Modulo 5: Authenticated Encryption e Analisi di AES-GCM

Basata su: Slide del corso, Boneh-Shoup (Cap. 9) e Analisi Visuale, registrazione lezione 18

Sommario

Questa dispensa analizza la necessità della Cifratura Autenticata (AE), dimostrando la vulnerabilità dei cifrari standard (CPA-sicuri) contro attacchi attivi. Vengono dissezionati gli attacchi su CBC e CTR e presentata la teoria formale dell'Authenticated Encryption secondo Boneh e Shoup. Infine, viene fornita un'analisi dettagliata dello standard AES-GCM, della sua costruzione algebrica su GF(2^{128}), dello schema Wegman-Carter e delle vulnerabilità catastrofiche legate al riutilizzo del nonce.

Indice

1 Il Fallimento della Sola Cifratura (CPA vs CCA)

Secondo Boneh e Shoup, la cifratura standard garantisce solo la sicurezza semantica contro attacchi a testo in chiaro scelto (IND-CPA). Tuttavia, in scenari reali (come la rete), l'avversario è **attivo**: può intercettare, modificare e iniettare messaggi. Senza integrità, la confidenzialità stessa crolla.

Di seguito analizziamo nel dettaglio due attacchi classici che sfruttano la malleabilità dei modi operativi.

1.1 Attacco 1: Bit-Flipping su CBC (Reindirizzamento della Porta)

Questo attacco è illustrato nelle immagini fornite (*Example 1* e relativi schemi). Dimostra come un attaccante possa modificare il comportamento di un tunnel IPsec cifrato senza conoscere la chiave.

1.1.1 Scenario e Analisi del Pacchetto

Consideriamo un pacchetto IPsec cifrato con AES in modalità **CBC (Cipher Block Chaining)**.

- Il pacchetto è composto da una serie di blocchi cifrati: $IV, C_0, C_1, C_2, \ldots$.
- Il primo blocco di testo in chiaro (P_0), corrispondente a C_0, contiene l'intestazione IP.
- All'interno di questa intestazione si trova la **Porta di Destinazione** (es. porta 80 per traffico Web).

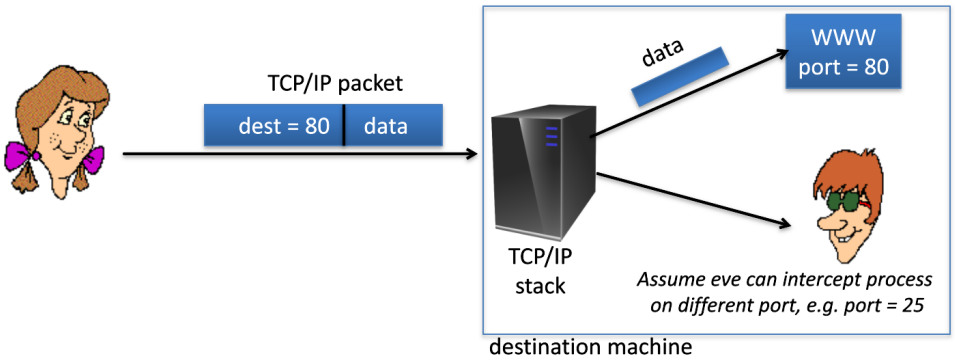

Figura 1: Situazione Iniziale attacco Bit-Flip

1.1.2 Meccanismo Matematico dell'Attacco

La decifratura del primo blocco in modalità CBC è definita dall'equazione:

$$P_0 = D_K(C_0) \oplus IV$$

L'attaccante vuole cambiare la porta da 80 a 25 (SMTP). Conosce la posizione dei byte della porta all'interno di P_0 (grazie agli standard IP) e conosce il valore attuale (80). Sfruttando la proprietà lineare dell'operazione XOR, l'attaccante calcola una maschera di modifica Δ:

$$\Delta = \text{Pad} \,\|\, (80 \oplus 25) \,\|\, \text{Pad}$$

Dove il "Pad" è costituito da zeri per tutti i bit che non devono essere modificati.

L'attaccante intercetta il pacchetto e modifica **solo** il Vettore di Inizializzazione (IV):

$$IV' = IV \oplus \Delta$$

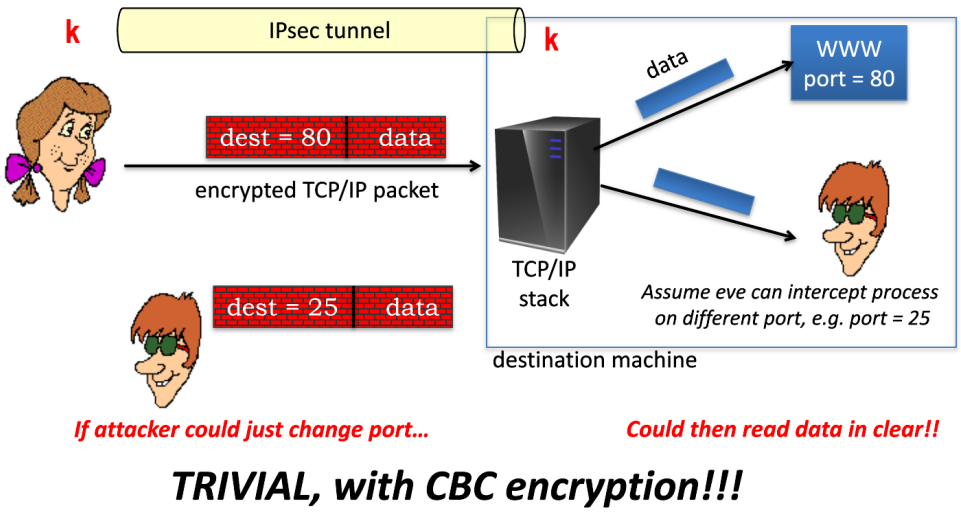

Figura 2: Cosa vuole fare l'attaccante

1.1.3 Esecuzione e Risultato

Il pacchetto manipolato $(IV', C_0, \dots)$ viene inviato al server di destinazione. Il server esegue la decifratura:

$$\begin{aligned} P_0' &= D_K(C_0) \oplus IV' \\ &= D_K(C_0) \oplus (IV \oplus \Delta) \end{aligned}$$

Poiché sappiamo che $D_K(C_0) \oplus IV = P_0$ (il plaintext originale), possiamo sostituire e ottenere:

$$P_0' = P_0 \oplus \Delta$$

Il risultato è che i bit corrispondenti alla porta 80 vengono "flippati" esattamente per diventare 25. Il resto del pacchetto rimane matematicamente valido e correttamente cifrato. Il server, vedendo la porta 25, inoltrerà il pacchetto al servizio di posta, completando l'attacco.

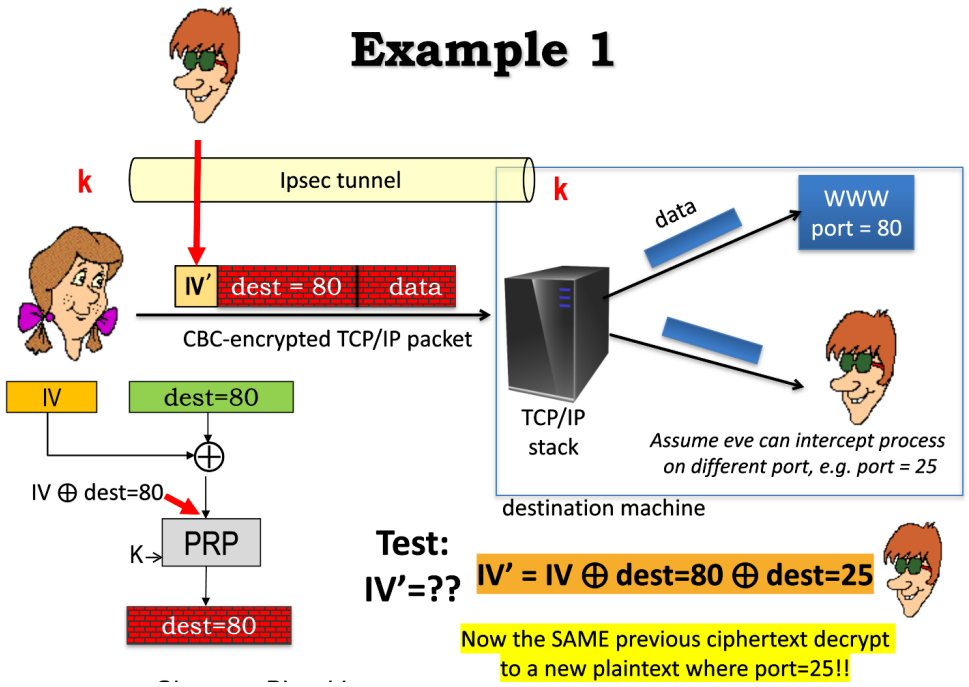

Figura 3: Situazione finale dell'attacco: l'attaccante ha cambiato effettivamente la porta

1.2 Attacco 2: Malleabilità di CTR e l'Oracolo TCP

Questo attacco (Slide Example 2) è più sottile e sfrutta un canale laterale (side-channel): il comportamento dello stack TCP/IP.

1.2.1 Scenario

Un terminale remoto invia singoli keystroke cifrati con AES in modalità **CTR (Counter Mode)**. Ogni pacchetto contiene: Header IP, Header TCP, Checksum (16 bit) e un singolo byte di Dati (il carattere digitato).

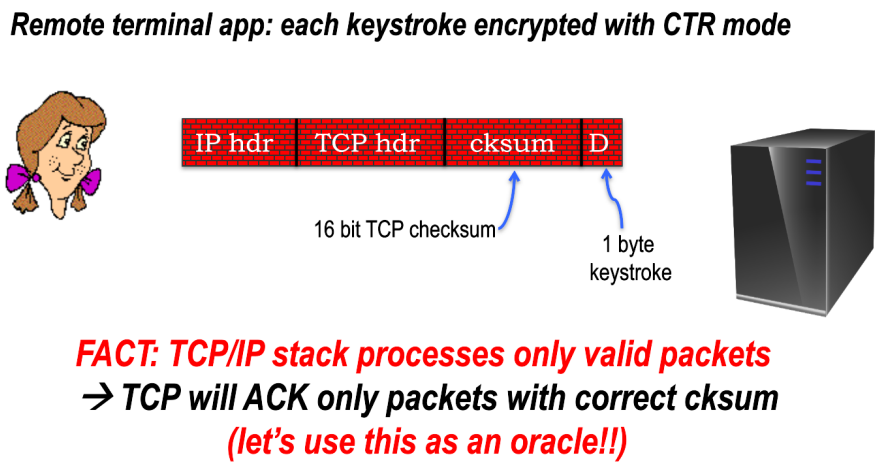

Figura 4: Situazione iniziale attacco di malleabilità su CTR

1.2.2 La Malleabilità di CTR

In modalità CTR, la cifratura è un semplice XOR tra il messaggio (M) e il keystream (S):

$$C = M \oplus S$$

Questa struttura rende il cifrario perfettamente malleabile a livello di bit (bit-flipping). Se l'attaccante inverte l'i-esimo bit del ciphertext C, inverte automaticamente l'i-esimo bit del plaintext M decifrato. Non c'è diffusione dell'errore come in CBC.

1.2.3 Esecuzione dell'Attacco

L'obiettivo è modificare il carattere 'd' (0x64) in 'c' (0x63). La differenza è solo l'ultimo bit.

1. **Modifica Dati:** L'attaccante intercetta C e inverte l'ultimo bit del byte cifrato dei dati. Ora il pacchetto decifrato conterrà 'c'.
2. **Il Problema del Checksum:** Il protocollo TCP usa un checksum per l'integrità. Modificando i dati, il checksum calcolato dal server sui dati decifrati non corrisponderà a quello presente nel pacchetto (ancora valido per 'd'). Il server scarterebbe il pacchetto.
3. **Correzione Cieca:** Il Checksum TCP è lineare rispetto allo XOR per piccole modifiche. L'attaccante applica la stessa modifica (o una compensazione calcolata) ai bit cifrati del campo Checksum nel pacchetto.
4. **L'Oracolo (ACK):** L'attaccante invia il pacchetto doppiamente modificato.
    - Se la modifica al checksum è errata → pacchetto scartato (silenzio).

- Se la modifica è corretta → il server accetta il pacchetto (payload 'c') e invia un **ACK**.

La ricezione dell'ACK conferma all'attaccante di aver forgiato un pacchetto valido, violando l'integrità del sistema.

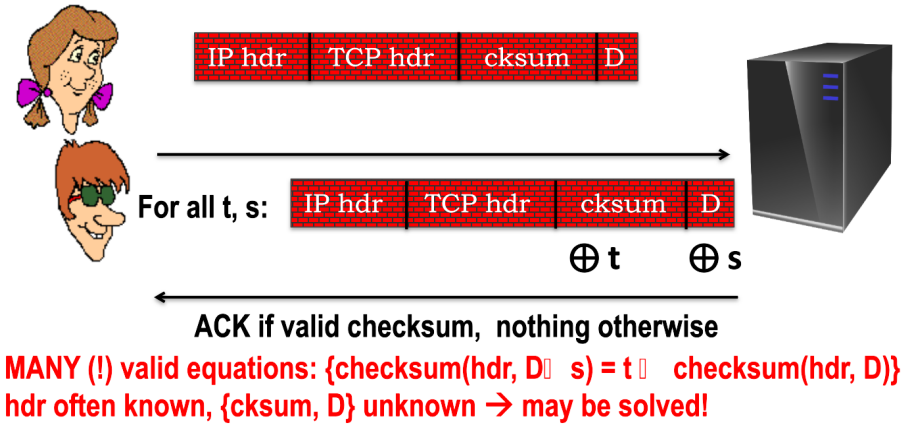

Figura 5: Situazione Finale attacco di malleabilità su CTR

La regola d'oro è quindi che la sicurezza CPA non può garantire la segretezza in caso di attacco attivo.
Possiamo usarla solo in uno dei due modi:

- Se il messaggio ha bisogno solo di integrità, ma non di confidenzialità → usare un **MAC**.
- Se servono entrambe, usare l' **Authenticated Encryption**.

2 Authenticated Encryption (AE): Definizioni Formali

Per risolvere questi problemi, Boneh e Shoup introducono il concetto di **Authenticated Encryption (AE)**. Un sistema AE non deve solo nascondere i dati, ma garantire che qualsiasi tentativo di modifica porti a un errore di decifratura.

2.1 Sintassi

Un sistema AE è una tripla di algoritmi (Gen, Enc, Dec):

- $Enc(K, M) \to C$: Cifra il messaggio M producendo un ciphertext C che include implicitamente o esplicitamente un *tag* di autenticazione.
- $Dec(K, C) \to M \cup \{\perp\}$: Decifra C. Se il ciphertext è valido, restituisce M. Se è stato modificato, restituisce il simbolo speciale $\perp$ (**REJECT**).

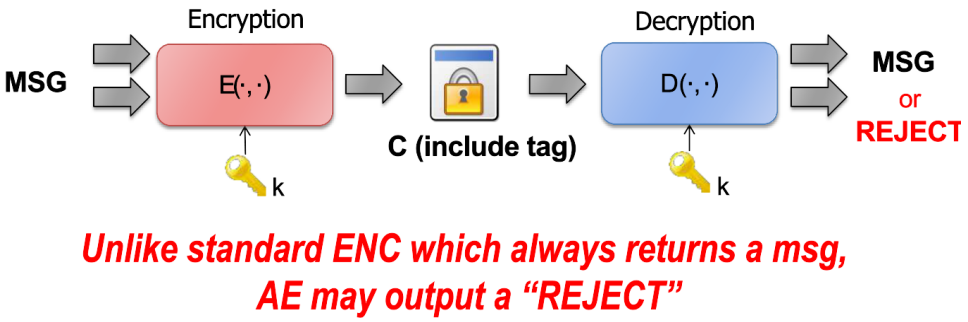

Figura 6: Schema di AE

AE è **fondamentalmente più sicuro** dei cifrari basici: effettua la decifratura *SOLO SE* il tag di autenticazione è valido → previene quindi gli attaccanti dall'effettuare attacchi di tipo CCA.

2.2 Sicurezza: Due Proprietà Fondamentali

Secondo Boneh-Shoup, un sistema AE sicuro deve soddisfare:

1. **CPA-Security (Confidenzialità):** Il ciphertext non deve rivelare informazioni sul plaintext.
2. **Ciphertext Integrity (CT-Integrity):** È computazionalmente impossibile per un attaccante creare un nuovo ciphertext C' (non generato precedentemente dall'oracolo di cifratura) tale che $Dec(K, C') \neq \perp$.

La combinazione di queste due proprietà garantisce la sicurezza contro attacchi a testo cifrato scelto (**CCA-Security**, oppure **IND-CCA**).

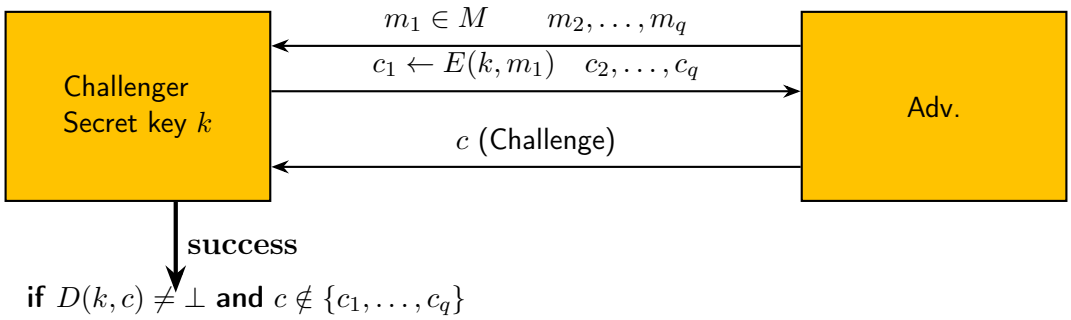

Figura 7: Gioco di Sicurezza per l'Integrità del Testo Cifrato (CT-Integrity).

2.3 AEAD: Associated Data

Negli standard moderni (come TLS), si usa **AEAD** (AE with Associated Data).

$$Enc(K, Nonce, AD, M) \rightarrow (C, Tag)$$

L'**Associated Data (AD)** (es. header di rete) non viene cifrato ma viene *autenticato*. Qualsiasi modifica all'AD invalida il tag e provoca il rigetto del messaggio.

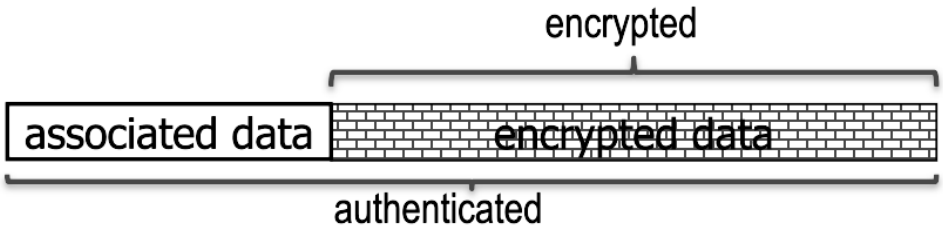

Figura 8: Schema AEAD

3 Scelte di Progettazione per AEAD

Quando si progetta uno schema AEAD, ci sono diverse scelte strutturali e funzionali da considerare (Slide 1327-1339).

3.1 Struttura (Structure)

- **Two-Layer (Due Passaggi):** Prima si cifra e poi si calcola il MAC (o viceversa). Esempio tipico: **AES-GCM** (Encrypt-then-MAC).
- **One-Layer (Un Passaggio):** Cifratura e autenticazione avvengono in un'unica passata sui dati. Esempio: **OCB (Offset Codebook Mode)**. Questo approccio è generalmente più veloce ma spesso coperto da brevetti (motivo per cui GCM è preferito).

3.2 Prestazioni (Performance)

- L'approccio "Encrypt-then-MAC" può richiedere due passaggi sui dati, risultando più lento.
- La **parallelizzabilità** è difficile da ottenere completamente, poiché il controllo di autenticità richiede di aver processato "tutti" i dati (funzione hash o MAC).
- **Streamability (Online Cipher):** È desiderabile che l'algoritmo possa processare i dati in un flusso continuo (one pass), senza dover bufferizzare l'intero messaggio prima di iniziare l'output.

3.3 Requisiti Funzionali

- **Posizionamento degli Associated Data:** Dove inserire gli AD? Prima, dopo o mischiati al testo cifrato?
- **Misuse Resistance (Resistenza all'uso improprio):** Quanto è robusto lo schema se l'IV viene riutilizzato per errore? (Vedi sezione successiva su GCM).

4 Costruzioni Generiche e AES-GCM

Boneh e Shoup analizzano diverse composizioni di Cifratura e MAC.

- **SSH (Encrypt-and-MAC):** Insicuro teoricamente (il MAC potrebbe rivelare info sul plaintext).
- **SSL (MAC-then-Encrypt):** Vulnerabile ad attacchi di padding oracle (es. Lucky13).
- **IPsec/Standard (Encrypt-then-MAC):** È la costruzione provabilmente sicura. Si cifra il messaggio ($C = E_K(M)$) e poi si calcola il MAC sul ciphertext ($T = MAC_{K'}(C)$).

4.1 AES-GCM: Panoramica e Costruzione ad Alto Livello

AES-GCM (Galois/Counter Mode) è una variante efficiente di *Encrypt-then-MAC*, progettata per alte prestazioni hardware.

- **Cifratura:** Usa AES-CTR (Counter Mode).
- **MAC:** Usa GHASH (basato su campi di Galois).

4.1.1 Costruzione ad Alto Livello (High Level Construction)

Le slide (1350-1434) illustrano passo dopo passo come viene costruito un messaggio cifrato e autenticato in GCM.

1. **Cifratura CTR:** Il messaggio viene diviso in blocchi (m_1, m_2). Ogni blocco viene messo in XOR con l'output di AES cifrato con un contatore incrementale $(IV|1, IV|2)$.

$$ct_i = m_i \oplus AES_K(IV||i)$$

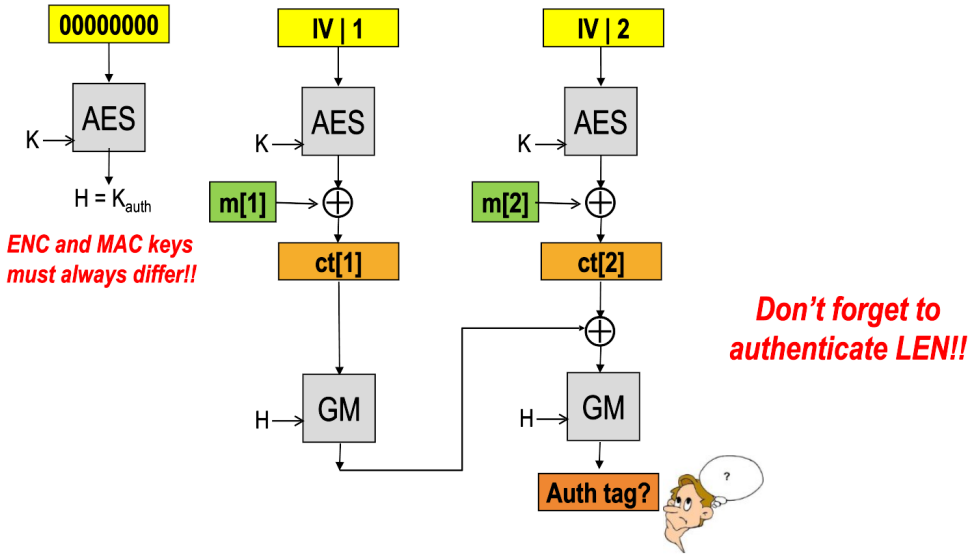

Figura 9: Cifratura con CTR

2. **Calcolo del MAC (GHASH):** I blocchi di testo cifrato (ct_1, ct_2) vengono passati attraverso la funzione GHASH. Questa è rappresentata nelle slide come una catena di XOR e Moltiplicazioni in GF(2^{128}) (GM - Galois Multiplication) con una chiave di hash H.

$$H = AES_K(0^{128})$$

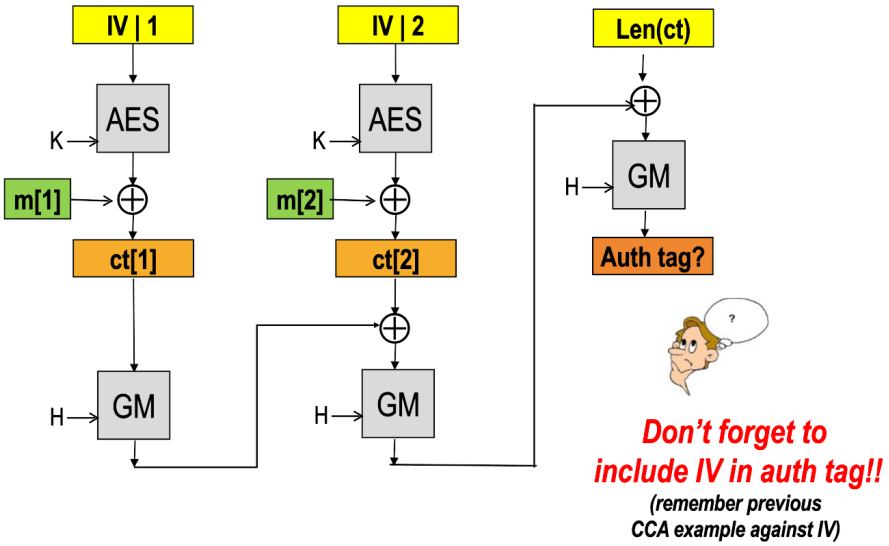

Figura 10: Calcolo del GHASH sui blocchi cifrati

3. **Autenticazione della Lunghezza:** Anche la lunghezza del ciphertext viene inclusa nel calcolo dell'hash per prevenire attacchi di estensione.

4. **Finalizzazione del Tag (Wegman-Carter):** L'output finale del GHASH viene messo in XOR con un "pad" cifrato generato da AES usando il contatore 0 ($IV|0$). Questo passaggio è fondamentale per rendere sicuro il MAC (vedi sezione successiva).

$$Tag = GHASH_H(...) \oplus AES_K(IV||0)$$

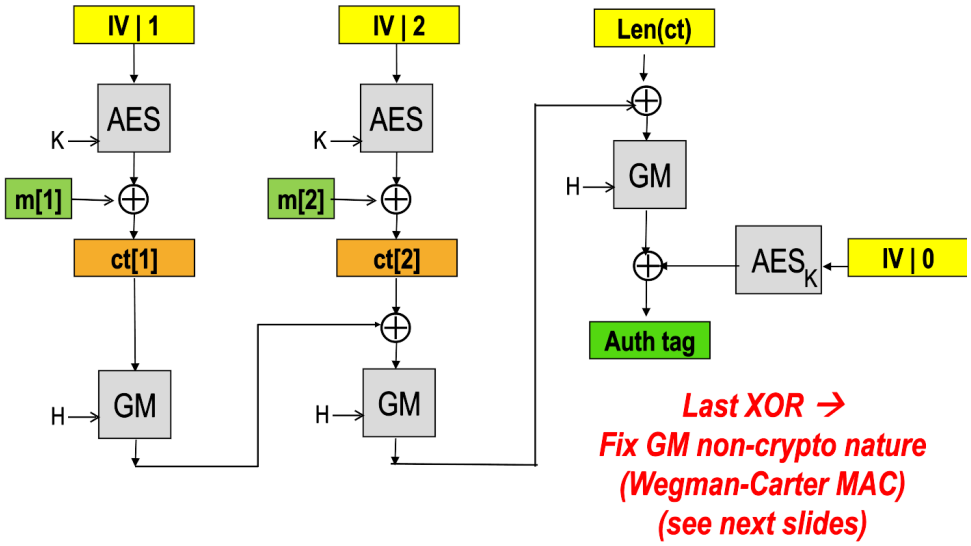

Figura 11: Finalizzazione con XOR del blocco contatore 0

5 Teoria del MAC di Wegman-Carter

Il meccanismo di autenticazione di GCM si basa sulla costruzione teorica di **Wegman-Carter**, che permette di costruire un MAC sicuro combinando una funzione hash non crittografica ma veloce (Universal Hash Function) con una funzione pseudo-casuale (PRF).

5.1 Universal Hash Function (UHF) e GHASH

GHASH non è una funzione hash crittografica (come SHA-256) resistente alle collisioni in senso forte. È una famiglia di funzioni hash "Universali".

- **Definizione:** Una famiglia di funzioni H_k è universale se, per ***un attaccante che NON conosce la chiave*** k, la probabilità di trovare una collisione ($H_k(M_1) = H_k(M_2)$) è bassa ($\leq 1/m$).
- **Proprietà:** Se si conosce la chiave, trovare collisioni è facile (GHASH è lineare!). Tuttavia, cambiando chiave, le collisioni cambiano in modo imprevedibile.

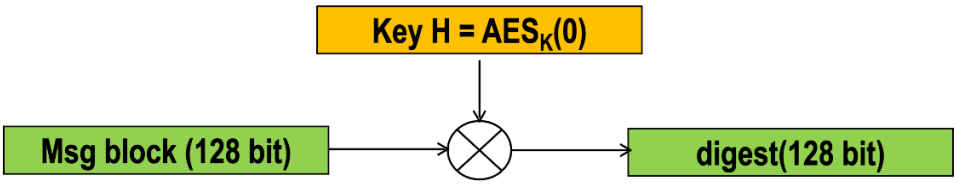

Figura 12: Costruzione di GHASH

5.2 La Costruzione Wegman-Carter

Wegman e Carter hanno dimostrato che si può ottenere un MAC sicuro combinando una UHF e una PRF:

$$MAC(K_1, K_2, M, Nonce) = UHF_{K_1}(M) \oplus PRF_{K_2}(Nonce)$$

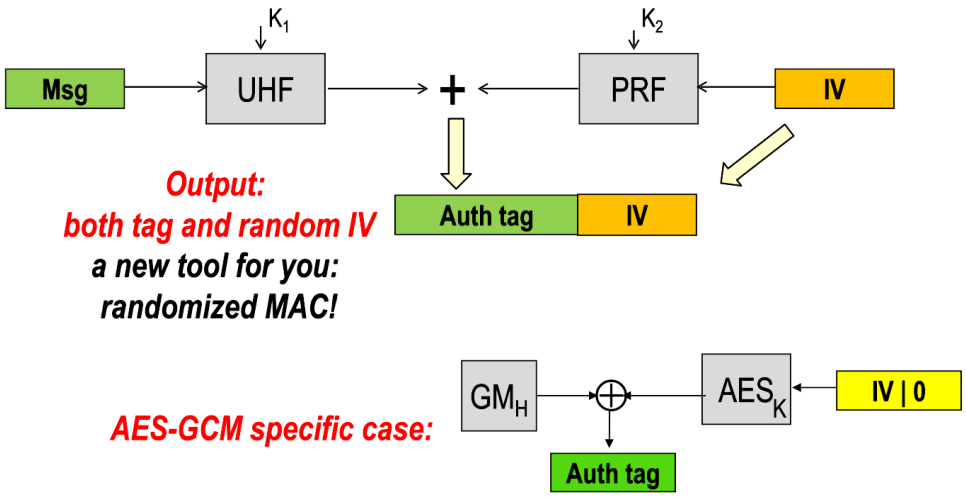

Figura 13: Schema della costruzione Wegman-Carter

In AES-GCM:

- **UHF:** È la funzione GHASH (moltiplicazione polinomiale).
- **PRF:** È il cifrario a blocchi AES.
- **Nonce:** È il blocco contatore 0 ($IV||0$).

Il risultato è un "Randomized MAC": ogni tag è unico e sicuro grazie alla mascheratura con il pad pseudo-casuale $AES_K(IV||0)$.

6 AES-GCM: Dettagli Algebrici

6.1 Struttura Algebrica: Il Campo GF(2^{128})

Il cuore dell'autenticazione in GCM è la funzione **GHASH**. Operazioni nel campo finito di Galois GF(2^{128}).

- Gli elementi sono polinomi di grado 127 a coefficienti binari $\{0, 1\}$.
- La somma è lo XOR bit a bit.
- La moltiplicazione ($\otimes$) è la moltiplicazione polinomiale modulo un polinomio irriducibile fissato $P(x) = x^{128} + x^7 + x^2 + x + 1$.
- Questa operazione è accelerata in hardware tramite l'istruzione `CLMUL` (Carry-less Multiplication).

6.2 Fasi Dettagliate

6.2.1 1. Cifratura (CTR)

$$C = AES\text{-}CTR_K(J_0, M)$$

Dove J_0 è il contatore iniziale derivato dall'IV.

6.2.2 2. Autenticazione (GHASH)

Si costruisce un polinomio i cui coefficienti sono i blocchi dell'Associated Data (A) e del Ciphertext (C), seguiti da un blocco con le lunghezze (Len).

$$S = GHASH_H(A, C) = A_1 H^{m+n+1} \oplus \ldots \oplus C_n H^2 \oplus Len \cdot H$$

Questa è essenzialmente una valutazione polinomiale nel punto H (schema di Horner).

6.2.3 3. Mascheramento del Tag

$$Tag = S \oplus AES_K(J_0)$$

Qui $AES_K(J_0)$ è il primo blocco del keystream CTR, che agisce come il pad one-time dello schema Wegman-Carter.

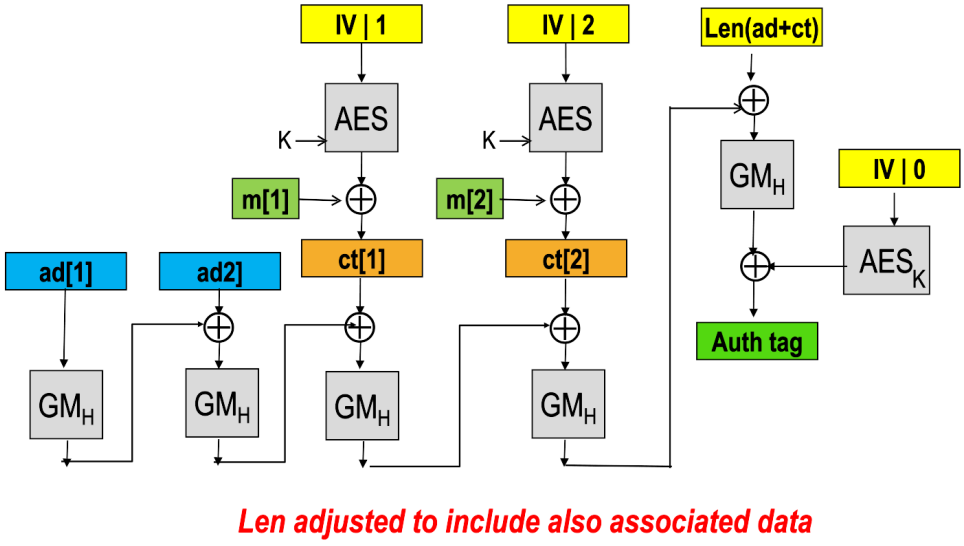

Figura 14: Schema finale AES-GCM

Come vediamo dalla figura, la prima cosa che viene fatta è aggiungere i blocchi relativi agli AD, quando finiamo gli AD iniziamo con il messaggio vero e proprio come spiegato poco prima. Alla fine, aggiungiamo l'ultimo blocco con $IV|0$ per creare il "Random MAC", ottenendo il TAG Valido per il messaggio

7 Il Disastro del Nonce Reuse in GCM

Boneh e Shoup, così come le slide, sottolineano che il riutilizzo del Nonce in GCM è molto più catastrofico che in CTR semplice.

Se un IV viene riutilizzato per due messaggi diversi:

1. **Perdita di Confidenzialità:** Come in CTR, $C_1 \oplus C_2 = M_1 \oplus M_2$.
2. **Perdita Totale dell'Integrità (Forgery):** Consideriamo due tag generati con lo stesso IV (e quindi stesso pad di mascheramento $E_K(J_0)$):

$$T_1 = GHASH_H(C_1) \oplus E_K(J_0)$$

$$T_2 = GHASH_H(C_2) \oplus E_K(J_0)$$

L'attaccante calcola lo XOR dei tag:

$$T_1 \oplus T_2 = GHASH_H(C_1) \oplus GHASH_H(C_2)$$

Grazie alla linearità di GHASH:

$$T_1 \oplus T_2 = GHASH_H(C_1 \oplus C_2)$$

Espandendo la definizione polinomiale di GHASH, questa diventa un'equazione polinomiale nella variabile H (la chiave di autenticazione).

Poiché l'attaccante conosce C_1, C_2, T_1, T_2, può risolvere l'equazione (trovare le radici del polinomio in GF(2^{128})) e recuperare **H**.

Conseguenza: Una volta noto H, l'attaccante può falsificare tag validi per qualsiasi messaggio a sua scelta. La sicurezza del sistema è annientata.

8 Conclusioni

- La cifratura non autenticata (CBC, CTR) è vulnerabile ad attacchi attivi banali.
- Si deve usare sempre AEAD.
- AES-GCM è performante ma fragile: il riutilizzo del Nonce rivela la chiave di autenticazione.
- Per ambienti dove l'unicità del Nonce non è garantita, si raccomandano modi resistenti al misuse come **AES-GCM-SIV** (RFC 8452).