Integrità ed Autenticazione

Riccardo Longo

Information Security

La sicurezza delle informazioni si può analizzare sulla base della triade **CIA**:

- **Confidentiality**: ha accesso solo chi ne ha diritto
  - Controllo degli accessi
  - **Cifratura**
- **Integrity**: i dati sono accurati e completi
  - Codici a correzione d'errore
  - **MAC** e **firme digitali**
- **Availability**: i dati sono disponibili quando servono
  - Anti-DOS/DDOS
  - Backup/repliche

Integrità

L'integrità vuole che:

- Il dato sia completo
- Non siano state fatte modifiche non autorizzate
- Non ci siano errori
- I dati siano consistenti

Autenticazione

Parallelamente all'integrità c'è l'**autenticazione**

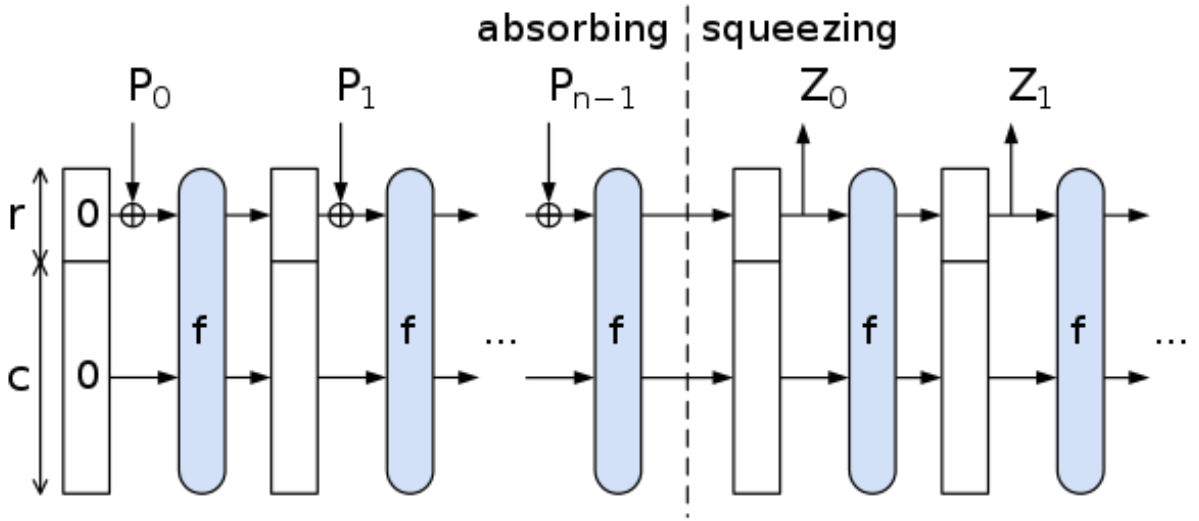
- Stabilire l'**origine** del dato
- Garantire la **genuinità** del dato
- **Non-ripudio**: responsabilità del dato
- **Deniable authentication**: i partecipanti riescono ad autenticare ma non possono dimostrarlo ad un esterno

Codici a Correzione d'Errore

- Contrastano gli **errori** di trasmissione e la **deperibilità** dei supporti di archiviazione
- Il dato è **codificato**:
  - Trasformazione in una parola del **codice**
  - Aggiunge informazioni di **controllo**
  - Permette di riconoscere e correggere gli errori
- La **decodifica** ricostruisce il messaggio originale, anche in presenza di errori (sotto una certa soglia)
- Usati a livello di **trasporto** e **archiviazione** dei dati

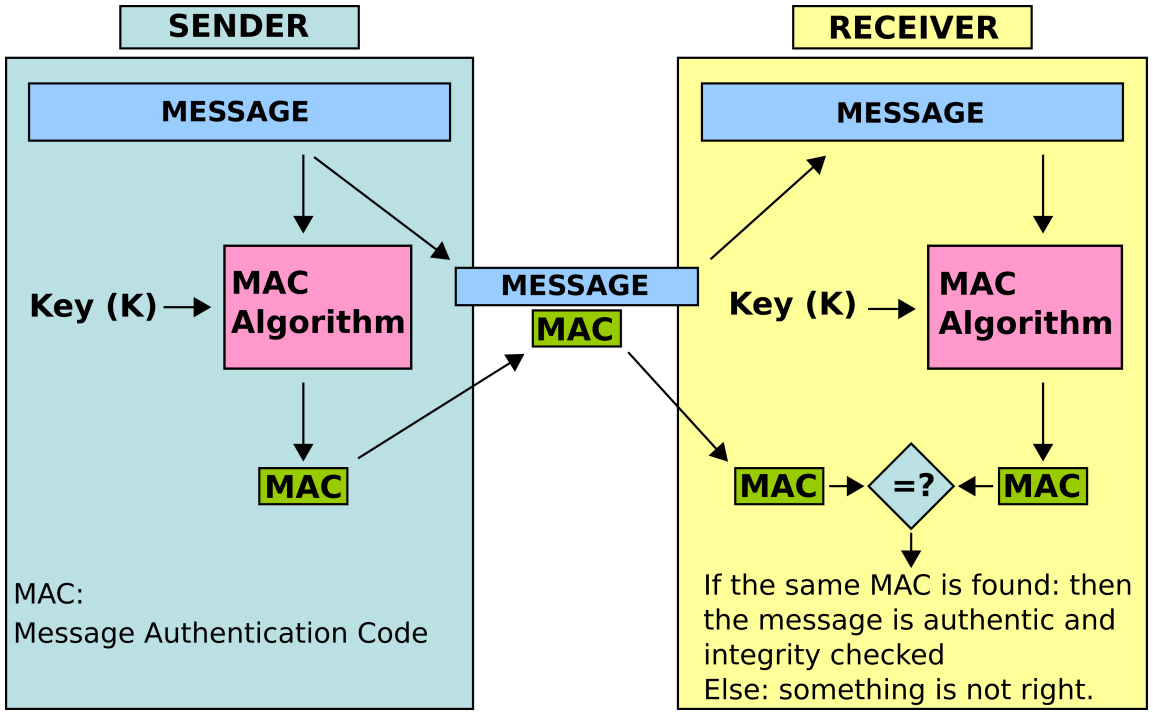
Hash Function

- Possono fornire integrità del messaggio
- **Hash Universale**:
  - Funzione scelta random da una famiglia
  - In media ogni output ha la stessa probabilità di essere scelto
- **Sponge**: modello di costruzione (SHA-3)



Message Authentication Codes (MAC)

- Informazioni aggiuntive ai messaggi (**tag**)
- Forniscono **integrità** e **autenticazione**



Struttura di un MAC

Un **MAC** è composto da tre algoritmi:

- **G**: generazione della chiave k a partire da un parametro di sicurezza
- **S**: firma (signing) ovvero generazione del tag t a partire dal messaggio x e della chiave k
- **V**: verifica del tag t e del messaggio x usando la chiave k

Deve soddifsfare la proprietà:

$$k \leftarrow \mathsf{G}(1^n) \implies \mathsf{V}(k, x, \mathsf{S}(k, x)) = \mathsf{OK}$$

Sicurezza

- Il tag non dev'essere contraffabile (**unforgeable**)
- Lo scenario è in genere di tipo **Chosen Message**
  - L'avversario può chiedere la creazione dei tag di messaggi a sua scelta
  - Deve riuscire a costruire un tag valido per un altro messaggio

$$k \leftarrow \mathsf{G}(1^n)$$

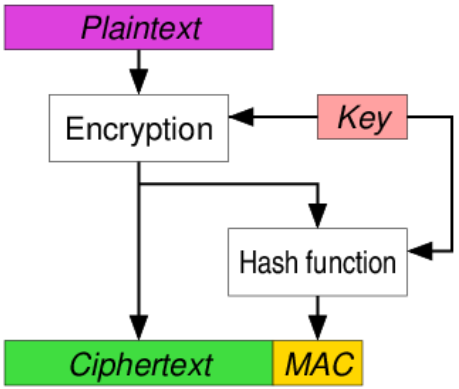
$$(x, t) \leftarrow \mathsf{A}^{(k,\cdot)}(1^n)$$

$$\tilde{x} \notin \mathsf{Query}(\mathsf{A}^{\mathsf{S}(k,\cdot)}, 1^n)$$

$$Pr\left[\mathsf{V}(k, \tilde{x}, \tilde{t}) = \mathtt{OK}\right] < \mathsf{negl}(n)$$

Authenticated Encryption

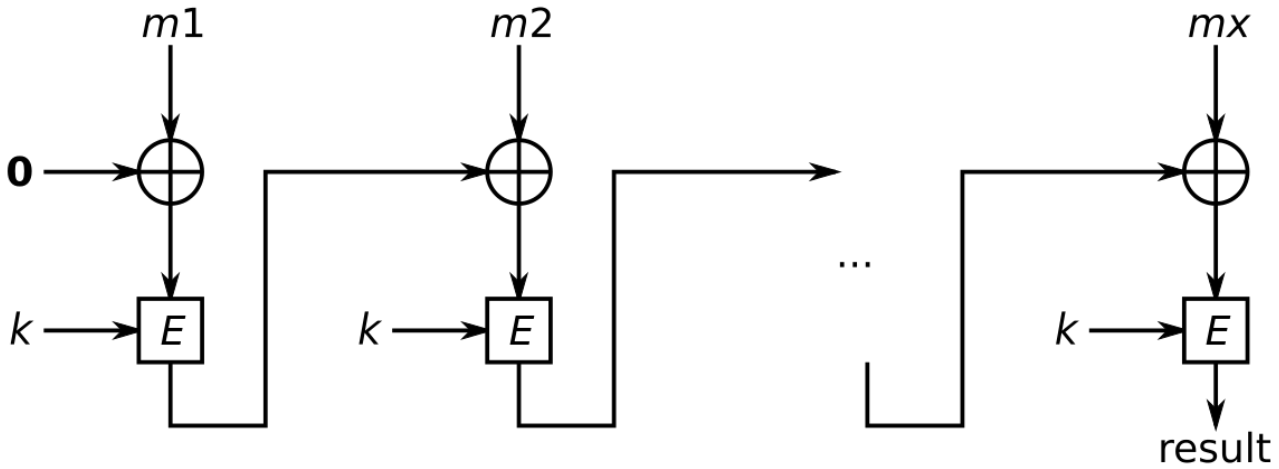
- Combinare **Confidenzialità** e **Integrità**
- Uniscono un cifrario con un MAC per fornire sicurezza contro **Adaptive Chosen Ciphertext Attack**
  - Il cifrario dev'essere **semanticamente sicuro** contro un **Chosen Plaintext Attack**
  - Il MAC dev'essere **unforgeable** contro un **Chosen Message Attack**



- La competizione **CAESAR** ha selezionato nel 2019 un portfolio di cifrari

  `https://competitions.cr.yp.to/caesar-submissions.html`

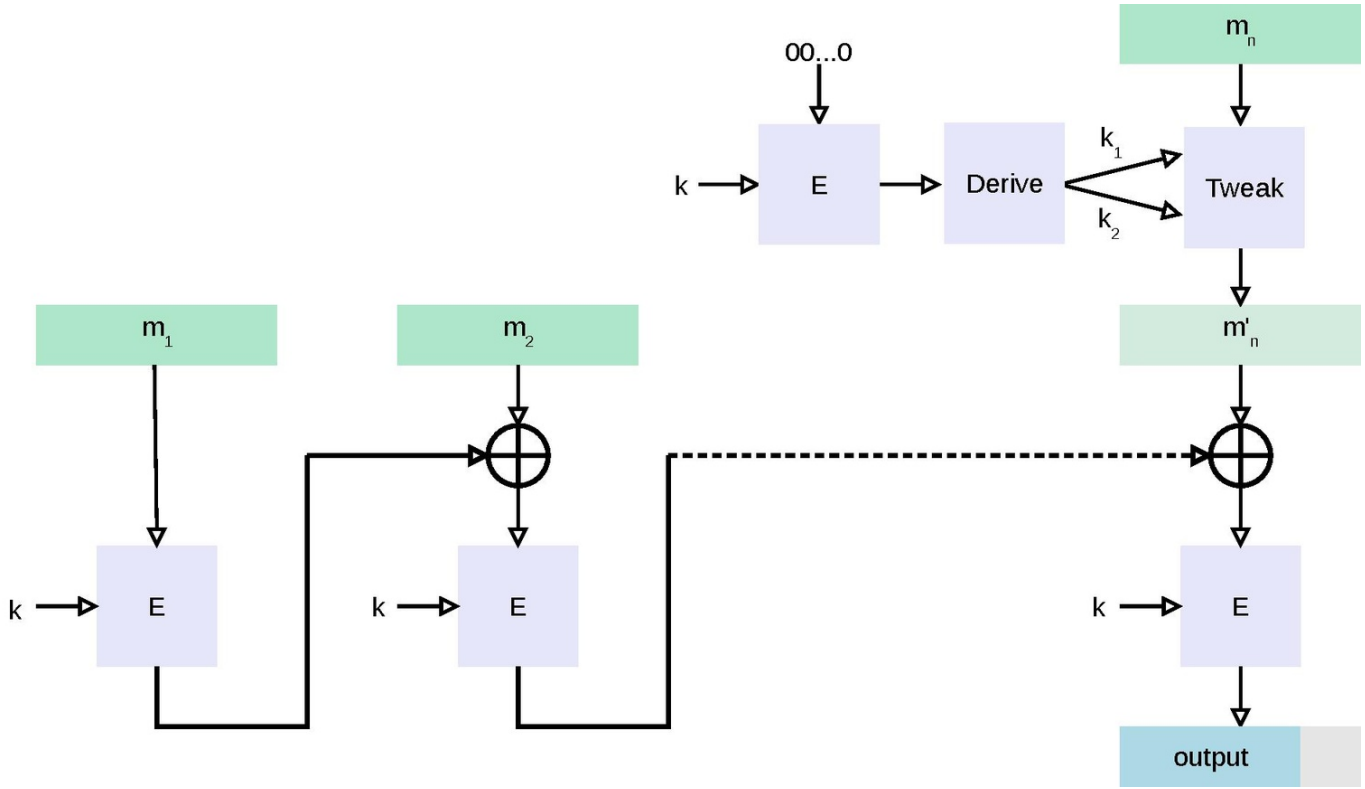
CBC-MAC



- Sicuro solo per messaggi a lunghezza prefissata: dati (m, t) e (m', t') si può costruire $m'' = m \| [(m'_1 \oplus t) \| m'_2 \| \dots \| m'_x]$ che ha lo stesso tag t'
- Se il messaggio viene poi cifrato con la stessa chiave è possibile alterare tutti i blocchi tranne l'ultimo
- Se l'**IV** è variabile o prevedibile si può contraffare un messaggio: $M_1 = P_1|P_2|\cdots \rightarrow M'_1 = (P_1 \oplus IV_1 \oplus IV_2)|P_2|\dots$

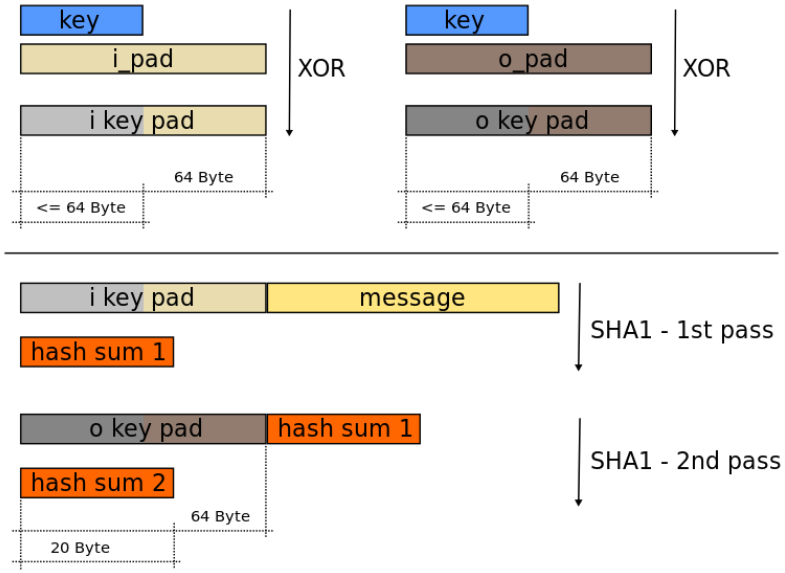
Cipher-based MAC (CMAC)

- Noto anche come One-key MAC (OMAC)
- Aggiusta le vulnerabilità di CBC-MAC



HMAC

- **Keyed-hash** o **hash-based** Message Authentication Code
- Basato su una qualsiasi hash crittografica sicura
- Progettato per proteggere contro **length-extension attacks** specifici di alcune hash (SHA-1, SHA-2)



$$\mathrm{HMAC}(K, m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ più lunga del blocco} \\ K & \text{altrimenti} \end{cases}$$

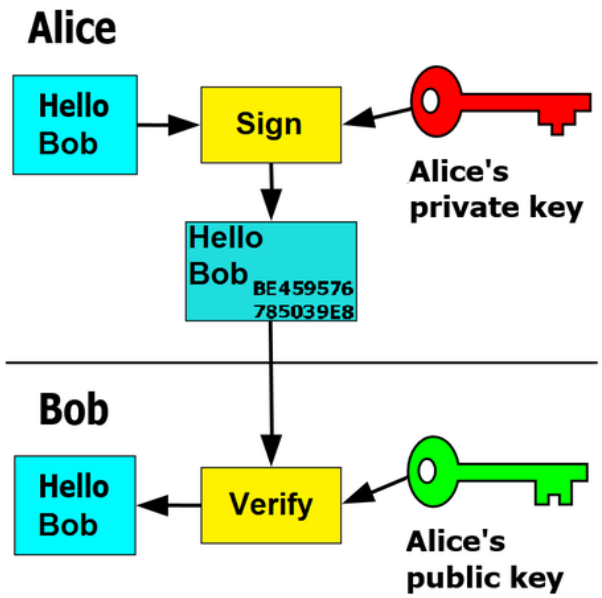
Poly1305

- Basata su una **Hash Universale** e un cifrario simmetrico
- Dimostrabilmente sicuro se il cifrario è sicuro
- Ottimizzato per efficienza e parallelizzabilità
- Fornisce 128 bit di sicurezza tramite 2 chiavi (k, r) ed un nonce n da 128 bit ognuno
- Il messaggio è diviso in q blocchi da 16 byte, e codificati in $c_i = m_i||100\ldots$, arrivando a blocchi finali di 17 byte
- Il tag è calcolato così (usando codifica *little-endian*):

$$\left(\sum_{i=0}^{q-1} c_i r^{q-i} \mod 2^{130} - 5\right) + \text{AES}(k, n) \mod 2^{128}$$

Firma Digitale

- Fornisce **autenticità** e **non-ripudio**
- Basata su crittografia **asimmetrica**
- Come i MAC è costituita da tre algoritmi:
  - **Generazione** di chiavi: coppia chiave **pubblica** e **privata**
  - **Firma**: usando la chiave **privata**
  - **Verifica**: usando la chiave **pubblica**



Sicurezza: Scenari d'attacco

- **Key Only**: l'attaccante conosce solo la chiave pubblica
- **Known Message**: l'attaccante conosce un certo numero di coppie messaggio-firma
- (**Adaptive**) **Chosen Message**: l'attaccante può scegliere un certo numero di messaggi da farsi firmare

Sicurezza: Risultati dell'attacco

- **Total Break**: l'attaccante recupera la chiave privata
- **Universal Forgery**: l'attaccante riesce a falsificare la firma di un qualunque messaggio
- **Selective Forgery**: l'attaccante riesce a falsificare la firma di un messaggio a sua scelta
- **Existential Forgery**: l'attaccante riesce a creare una coppia messaggio-firma (valida) che non conosceva già

ElGamal Signature Scheme

- **Generazione chiavi**: p primo grande, g *generatore* del gruppo moltiplicativo $\mathbb{Z}_p^*$
  - Genera $1 < x < p-2$ random, SK $= x$
  - PK $= y = g^x$, (p, g) parametri pubblici, condivisibili
- **Firma**: $1 < k < p-1$ random tale che $\gcd(k, p-1) = 1$, messaggio m codificato come intero
  - $r = g^k$, $s = (m - xr)k^{-1} \mod p-1$, se $s = 0$ cambia k
  - La firma è (r, s)
- **Verifica** la firma è valida se:
  - $0 < r < p$ e $0 < s < p-1$
  - $g^m \equiv y^r r^s \pmod p$

$$y^r r^s \pmod p \equiv (g^x)^r (g^k)^s \equiv g^{xr} g^{ks} \equiv g^{xr + k(m-xr)k^{-1}} \equiv g^{xr+m-xr} \equiv g^m$$

Existential Forgery di ElGamal

- A un parametro:
  - $1 < e < p-1$ random
  - (r, s) è una **firma valida** per $m = es \pmod{p-1}$ se $r = g^e y \pmod{p}$, $s = -r \pmod{p-1}$

$$g^m \equiv g^{es} \equiv g^{-er} \equiv y^r g^{-er} y^{-r} \equiv y^r (g^e y)^{-r} \equiv y^r r^s$$

- A due parametri:
  - $1 < e, v < p-1$ random tali che $\gcd(v, p-1) = 1$
  - Se $r = g^e y^v \pmod{p}$ e $s = -rv^{-1} \pmod{p-1}$, allora (r, s) è una **firma valida** per il messaggio $m = es \pmod{p-1}$

$$g^m \equiv g^{es} \equiv g^{-e\frac{r}{v}} \equiv y^r g^{-e\frac{r}{v}} y^{-r} \equiv y^r g^{-e\frac{r}{v}} y^{-v\frac{r}{v}} \equiv y^r (g^e y^v)^{-\frac{r}{v}} \equiv y^r r^s$$

DSA

- Standard NIST, variante di ElGamal
- Non firma direttamente il messaggio, ma l'hash
  - Più efficiente
  - Assicura l'integrità del messaggio intero
  - Previene Existential Forgery
- Parametri:
  - q primo di $N \geq 160$ (256) bit
  - p primo di $L \geq 1024$ (2048) bit, con $p - 1$ multiplo di q
  - g generatore del sottogruppo di ordine q di $\mathbb{Z}_p$, $g = h^{\frac{p-1}{q}}$, solitamente $h = 2$

DSA: algoritmi

- **Generazione chiavi**:
  - Genera $0 < x < q$ random, SK $= x$
  - PK $= y = g^x \mod p$
- **Firma**: $1 < k < q$ random
  - $r = (g^k \mod p) \mod q$, $s = (H(m) + xr)k^{-1} \mod q$
  - Se $sr = 0$ cambia k, altrimenti la firma è (r, s)
- **Verifica** la firma è valida se:
  - $0 < r < q$ e $0 < s < q$
  - $r = (g^{u_1} y^{u_2} \mod p) \mod q$, con:
    - $w = s^{-1} \mod q$
    - $u_1 = H(m) \cdot w \mod q$
    - $u_2 = r \cdot w \mod q$

ECDSA

- Variante di DSA con le curve ellittiche
- Chiavi molto più **corte**: 256 bit vs 2048 bit
- Firma di lunghezza simile: ~ 512 bit
- I parametri vanno a definire la curva ellittica ed il **punto base** G, equivalente al generatore g
- Sia per DSA che ECDSA k è cruciale per la sicurezza, dev'essere:
  - **Unico**
  - **Segreto**
- Un riuso o informazioni anche parziali su k portano alla **rottura completa** della firma:
  - Nel 2010 compromessa la firma della **PlayStation3**
  - Nel 2013 violati molti **wallet Bitcoin**

EdDSA

- Variante di ECDSA mirata a prevenire vulnerabilità
- Basato sulla firma di **Schnorr**
- Usa particolari curve chiamate **Twisted Edwards curves**
  - Molto **efficienti**
  - Resistenti a **side channel** poiché usano le stesse formule sia per la somma che il raddoppio di punti della curva
- Il parametro k non random ma **deterministico**:
  - Viene usata una funzione pseudorandom partendo dalla chiave privata e dal messaggio
  - Assicura valori diversi per ogni messaggio
- **Ed25519** variante ottimizzata che offre 128 bit di sicurezza usando la curva di equazione

$$-x^2 + y^2 = 1 - \frac{121665}{121666}x^2y^2$$

sul campo $\mathbb{F}_q$, $q = 2^{255} - 19$