5 Funzioni Hash Crittografiche

In questo capitolo introdurremo le funzioni hash crittografiche e analizzeremo il loro utilizzo per assicurare l'integrità dei dati.

5.1 Funzioni Hash

Sia $\mathcal{X}$ un insieme di stringhe binarie di lunghezza arbitraria e $\mathcal{Y}$ un insieme di stringhe arbitrarie di lunghezza fissa, generalmente di circa 160 bit.

Definizione 5.1. (Funzione Hash).
Una **funzione hash** (priva di chiave) è una funzione $h : \mathcal{X} \longrightarrow \mathcal{Y}$, dove
1. $\mathcal{X}$ è l'insieme dei possibili **messaggi**
2. $\mathcal{Y}$ è l'insieme dei **sunti dei messaggi**.

Supponiamo che $y = h(x)$ sia conservato in un posto sicuro, mentre x non lo sia. Se il messaggio x è alterato in x' e vale che $y' \neq y$, dove $y' = h(x')$, allora è sufficiente stabilire che x è stato alterato semplicemente calcolando $y' = h(x')$ e valutando che $y' \neq y$.

Un vantaggio nell'uso della funzione hash è che i dati da conservare in un posto sicuro occupano poca memoria.

5.2 MAC

Definizione 5.2. (MAC).
Una **famiglia hash con chiave**, ovvero un **codice di autenticazione dei messaggi (MAC)**, è una quadrupla $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ di insiemi non vuoti in cui
1. $\mathcal{X}$ è l'insieme dei possibili **messaggi**;
2. $\mathcal{Y}$ è l'insieme finito dei possibili **sunti dei messaggi** (o **etichette di autenticazione**);
3. $\mathcal{K}$ è l'insieme delle **chiavi**;
4. Per ogni $K \in \mathcal{K}$ esiste $h_K \in \mathcal{H}$, dove $h_K : \mathcal{X} \longrightarrow \mathcal{Y}$ è una **funzione hash**.

L'utilizzo di MAC avviene come segue: due utenti A e B scelgono segretamente una chiave K che individua una funzione hash h_K.
Se x è un messaggio e $y = h_K(x)$ il suo sunto, allora la coppia (x, y) può essere trasmessa da A attraverso un canale insicuro. Una volta che l'utente B riceve (x, y), calcola $h_K(x)$ e verifica che $y = h_K(x)$. Se questa condizione è soddisfatta, allora B è fiducioso che x non è stato alterato.

Si noti che a differenza delle funzioni hash prive di chiave, nel caso dei MAC entrambi x e y possono essere trasmessi attraverso un canale insicuro.
Le funzioni hash prive di chiave possono essere pensate come MAC in cui $|\mathcal{K}| = 1$.

- A differenza di $\mathcal{Y}$, l'insieme $\mathcal{X}$ dei messaggi può essere anche infinito. Nel caso in cui $\mathcal{X}$ sia finito, la funzione hash si dice **funzione di compressione**.

  In seguito, assumiamo che $$|\mathcal{X}| \geq |\mathcal{Y}|$$
  e, quando specificato, $$|\mathcal{X}| \geq 2\,|\mathcal{Y}|\,.$$
- Se $|\mathcal{X}| = N$ e $|\mathcal{Y}| = M$, denotato con $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$ l'insieme di tutte le funzioni da $\mathcal{X}$ in $\mathcal{Y}$, allora $\left|\mathcal{F}^{\mathcal{X},\mathcal{Y}}\right| = M^N$. Una qualsiasi sottoinsieme $\mathcal{F}$ di $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$ verrà denotato con (N, M)**-famiglia hash**.
- Una coppia $(x, y) \in \mathcal{X} \times \mathcal{Y}$ si dice **valida**, se $y = h_K(Y)$. Uno dei punti su cui focalizzeremo il nostro studio sarà quello di prevenire la costruzione di coppie valide da parte di un avversario.

5.3 Sicurezza delle Funzioni Hash

La sicurezza di una funzione hash priva di chiave $h : \mathcal{X} \longrightarrow \mathcal{Y}$ è stabilita **principalmente** rispetto ai tre seguenti problemi:

Problema 5.3. **(Immagine Inversa).**
Dati una funzione hash $h : \mathcal{X} \longrightarrow \mathcal{Y}$ e $y \in \mathcal{Y}$, determinare $x \in \mathcal{X}$ tale che $y = h(x)$.

Una funzione hash per cui il **Problema dell'Immagine Inversa** non può essere risolto in modo efficiente si dice **one-way**.

Problema 5.4. **(Seconda Immagine Inversa).**
Dati una funzione hash $h : \mathcal{X} \longrightarrow \mathcal{Y}$ e $x \in \mathcal{X}$, determinare $x' \in \mathcal{X}$ tale che $x \neq x'$ e $h(x') = h(x)$.

Una funzione hash per cui il **Problema della Seconda Immagine Inversa** non può essere risolto in modo efficiente si dice **resistente alla seconda immagine inversa**.

Problema 5.5. **(Collisione).**
Dati una funzione hash $h : \mathcal{X} \longrightarrow \mathcal{Y}$, determinare $x, x' \in \mathcal{X}$ tale che $x \neq x'$ e $h(x') = h(x)$.

Una funzione hash per cui il **Problema della Collisione** non può essere risolto in modo efficiente si dice **resistente alla collisione**.

Analizzeremo queste problematiche nell'ambito del Modello dell'Oracolo Casuale spiegato qui di seguito.

5.4 Il Modello dell'Oracolo Casuale

Figura 5.1: L'Oracolo di Delfi è l'oracolo più famoso della religione greca del periodo arcaico

Nella realtà esistono diverse costruzioni basate su funzioni hash che non possono essere provate sicure sul solo fatto che la funzione hash rispetti i requisiti di sicurezza sopra descritti. Un approccio, seppur controverso, è l'introduzione di modelli idealizzati in cui provare la sicurezza di funzioni hash. Esso fornisce un'evidenza (non una dimostrazione!) della sicurezza delle funzioni hash.
L'obiettivo di questa sezione è quello di acquisire il concetto di funzione hash 'ideale'.

Se una funzione hash è ben definita $h : \mathcal{X} \longrightarrow \mathcal{Y}$, l'unico modo per determinare il valore che h assume in corrispondenza del messaggio x è quello di calcolare $h(x)$. Questo deve rimanere vero anche se molti altri valori $h(x_1), ..., h(x_n)$ sono stati calcolati. Forniamo un esempio in cui ciò non si verifica.
Per esempio, fissato un intero $n \geq 2$, per ogni $a, b \in \mathbb{Z}_n$ si consideri la funzione bilineare

$$h : \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathbb{Z}_n, (x, y) \longmapsto ax + by.$$

Supponiamo di conoscere $h(x_1, y_1) = z_1$ e $h(x_2, y_2) = z_2$. Per ogni $r, s \in \mathbb{Z}_n$, vale che

$$h(r(x_1, y_1) + s(x_2, y_2)) = rh(x_1, y_1) + sh(x_2, y_2) = rz_1 + sz_2$$

Quindi, dalla conoscenza dei valori assunti da h in z_1 e z_2 ricaviamo il valore assunto da h in diversi altri punti senza necessariamente calcolare il valore di h in questi ulteriori punti.

Nel 1993 Bellare e Rogaway introdussero il **Modello dell'Oracolo Casuale**, che è un modello matematico che fornisce una funzione hash 'ideale'. Esso non esiste nella realtà, è stato introdotto per fornire una misura di confidenza nella robustezza della funzione hash progettata. Nella realtà, quando si progetta una funzione hash h, si testa la sicurezza di h nel modello dell'oracolo casuale per avere una **evidenza** della sua sicurezza.


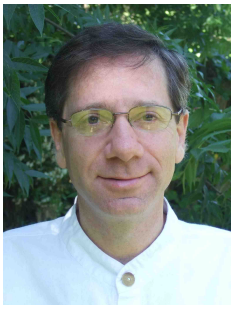

Figura 5.2: Mihir Bellare (1962) e Phillip Rogaway (1962)

Il Modello dell'Oracolo Casuale è sintetizzato come segue:

1. L'oracolo sceglie in modo casuale una funzione hash in $\mathcal{F}^{\mathcal{X},\mathcal{Y}}$.
2. Gli utenti (avversari inclusi) non hanno a disposizione una formula o un algoritmo che permetta loro di calcolare il valore di h in x, possono chiedere all'oracolo il valore $h(x)$.
3. I meccanismi interni all'oracolo non sono noti (scatola nera).
4. Le richieste sono private: nessuno sa la richiesta fatta da un utente, nessuno sa se un'utente ha fatto una richiesta.
5. Le risposte date dall'oracolo sono coerenti. Se in seguito alla richiesta x l'oracolo fornisce $h(x)$, questo valore viene memorizzato in una tabella. Successivamente, se viene fatta la stessa richiesta x anche da utenti diversi, l'oracolo recupera $h(x)$ dalla tabella senza calcolarlo nuovamente. Quindi, i valori vengono prima cercati nella tabella, e se questi non compaiono vengono calcolati e aggiunti.

Quindi, l'oracolo appare come una tabella di numeri casuali. Così per un utente che avanza all'oracolo la richiesta x i valori assumibili da $h(x)$ sono tutti equiprobabili e indipendenti. Pertanto, vale il seguente teorema di immediata dimostrazione.

Teorema 5.6. *Siano $h \in \mathcal{F}^{\mathcal{X},\mathcal{Y}}$ e $\mathcal{X}_0 \subseteq \mathcal{X}$ e supponiamo che nel modello dell'oracolo casuale siano stati determinati $h(x)$ per tutti gli $x \in \mathcal{X}_0$. Allora per ogni $x \in \mathcal{X} - \mathcal{X}_0$ e $y \in \mathcal{Y}$ vale che $\boldsymbol{P}[h(x) = y] = 1/M$.*

5.5 Algoritmi nel Modello dell'Oracolo Casuale

Analizziamo la complessità dei problemi di Immagine Inversa, Seconda Immagine Inversa e Collisione nel Modello dell'Oracolo Casuale. Per fare ciò introduciamo i seguenti concetti:

Definizione 5.7. **(Algoritmo Randomizzato).**
Gli **algoritmi randomizzati** sono algoritmi che fanno scelte casuali durante la loro esecuzione.

Definizione 5.8. **(Algoritmo Las Vegas).**
Un **algoritmo Las Vegas** è un algoritmo randomizzato che termina o fornendo una risposta corretta o che fallisce nel del dare una risposta.

Definizione 5.9. **(Probabilità di successo di un algoritmo).**

- Un algoritmo randomizzato ha probabilità di successo ε, dove $0 \leq \varepsilon < 1$, nel peggiore dei casi, se la probabilità di fornire una risposta corretta ad una istanza di input è maggiore o uguale a ε.
- Un algoritmo randomizzato ha probabilità di successo ε, dove $0 \leq \varepsilon < 1$, nella media dei casi, se la probabilità di fornire una risposta corretta, nella media delle delle istanze di input di fissata lunghezza è maggiore o uguale a ε.

Si noti che nel secondo caso, la probabilità di successa di fornire una risposta corretta ad una istanza di input può essere minore di ε.

Definizione 5.10. **((ε, q)-algoritmo).**
Nel modello dell'oracolo un **(ε, q)-algoritmo** è un algoritmo Las Vegas che ha probabilità di successo ε nella media dei casi, in cui il numero delle richieste all'oracolo (ovvero di valutazioni di h) è al più q.
La probabilità di successo ε è la media fatto su tutte le possibili scelte casuali di $h \in \mathcal{F}^{\mathcal{X},\mathcal{Y}}$, e su tutte le possibili scelte casuali di $x \in \mathcal{X}$ e $y \in \mathcal{Y}$, se x e/o y fanno delle istanze di input.

Algoritmo 5.11. **(Ricerca dell'immagine inversa (h, y, q)).**

Scelto $\mathcal{X}_0 \subseteq \mathcal{X}$ tale che $|\mathcal{X}_0| = q$.

for $x \in \mathcal{X}_0$ **do** $\begin{cases} \textbf{if } h(x) = y \\ \textbf{then return } (x) \end{cases}$

return (insuccesso)

Vale il seguente

Teorema 5.12. *L'**Algoritmo 5.11** è un* $\left(1 - \left(1 - \frac{1}{M}\right)^q, q\right)$*-algoritmo.*

Dimostrazione. Siano $y \in \mathcal{Y}$ e $\mathcal{X}_0 = \{x_1, ..., x_q\}$ e si denoti con $E_i(y)$ l'evento $h(x_i) = y$. Segue dal **Teorema 5.6**, che gli eventi $E_1(y), ..., E_q(y)$ sono indipendenti ed equiprobabili, quindi che vale $\boldsymbol{P}[E_i(y)] = 1/M$. Dalle **Leggi di De Morgan** e dall'indipendenza, si ha

$$\boldsymbol{P}\left[E_i(y)\right] = \boldsymbol{P}\left[\bigcup_{i=1}^{q} E_i(y)\right] = 1 - \boldsymbol{P}\left[\bigcap_{i=1}^{q} E_i(y)^{\mathcal{C}}\right] = 1 - \left(1 - \frac{1}{M}\right)^q.$$

La probabilià per un qualsiasi y è costante e quindi coincide la probabilità di successo mediata su tutti gli y in $\mathcal{Y}$, infatti vale

$$\boldsymbol{P} = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \boldsymbol{P}\left[\bigcup_{i=1}^{q} E_i(y)\right] = \frac{1}{|\mathcal{Y}|} |\mathcal{Y}| \left(1 - \left(1 - \frac{1}{M}\right)^q\right) = \boldsymbol{P}\left[E_i(y)\right].$$

□

Per q piccolo rispetto ad M, la probabilià media di successo è circa

$$1 - \left(1 - \frac{1}{M}\right)^q = 1 - \left(1 - \frac{1}{M}\right)^{-M\left(-\frac{q}{M}\right)} \simeq 1 - e^{\left(-\frac{q}{M}\right)} \simeq \frac{q}{M}.$$

Algoritmo 5.13. **(Ricerca della seconda immagine inversa (h, x, q)).**

$y \longleftarrow h(x)$

Scelto $\mathcal{X}_0 \subseteq \mathcal{X} - \{x\}$ tale che $|\mathcal{X}_0| = q - 1$.

for $x_0 \in \mathcal{X}_0$ **do** $\begin{cases} \textbf{if } h(x_0) = y \\ \textbf{then return } (x_0) \end{cases}$

return (insuccesso)

Vale il seguente

Teorema 5.14. *L'**Algoritmo 5.13** è un* $\left(1-\left(1-\frac{1}{M}\right)^{q-1}, q-1\right)$*-algoritmo.*

Dimostrazione. Siano $x \in \mathcal{X}$ e $\mathcal{X}_0 = \{x_1, ..., x_{q-1}\} \subseteq \mathcal{X} - \{x\}$, si denoti con $E_i(x)$ l'evento $h(x_i) = h(x)$. Come nel teorema precedente, gli eventi $E_1(x), ..., E_{q-1}(x)$ sono indipendenti ed equiprobabili, quindi vale che $\boldsymbol{P}[E_i(x)] = 1/M$ e

$$\boldsymbol{P}\left[\bigcup_{i=1}^{q-1} E_i(x)\right] = 1 - \left(1 - \frac{1}{M}\right)^{q-1}.$$

La probabilià per un qualsiasi x è costante e quindi coincide la probabilità di successo mediata su tutti gli x in $\mathcal{X}$.

□

Per q piccolo rispetto ad M, la probabilià media di successo è circa

$$1 - \left(1 - \frac{1}{M}\right)^{q-1} \simeq \frac{q-1}{M}.$$

Algoritmo 5.15. **(Ricerca di collisioni (h, q)).**

Scelto $\mathcal{X}_0 \subseteq \mathcal{X}$ tale che $|\mathcal{X}_0| = q$.

for $x \in \mathcal{X}_0$ **do** $y_x \longleftarrow h(x)$

if $y_x = y_{x'}$, $x' \neq x$, **then return** (x, x') **else return** (insuccesso)

L'**algoritmo 5.15** è analizzato attraverso il celebre **problema dei compleanni**, introdotto da Von Mises:

"Se in una stanza sono presenti q persone, qual è la probabilità che almeno due tra i presenti abbiano il compleanno nello stesso giorno?"

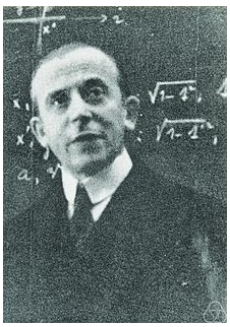

Figura 5.3: Richard von Mises (1883 – 1953)

Il **problema dei compleanni** può essere considerato un caso particolare del seguente problema:

Si estraggono con restituzione q palline da un'urna che ne contiene M, numerate da 1 a M. Qual è la probabilità che non si estraggano palline con lo stesso numero?
Evidentemente,

$$\Omega_q = \{(x_1, x_2, \cdots, x_q) : x_i = 1, 2, \cdots, M \quad (i = 1, 2, \cdots, q)\}$$

rappresenta tutti i possibili risultati nell'estrazione delle q palline, allora si ha $|\Omega_q| = M^q$. Si chiede di calcolare la probabilità dell'evento

$$A_q := \{(x_1, x_2, \cdots, x_q) : x_i \neq x_j \quad (i \neq j)\}.$$

$N(A_q) = D_{M,q}$ se $q \leq M$, $N(A_q) = 0$ se $q > M$, sicché, se $q \leq M$,

$$\mathbf{P}(A_q) = \left(1 - \frac{1}{M}\right)\left(1 - \frac{2}{M}\right)\cdots\left(1 - \frac{q-1}{M}\right).$$

Allora la probabilità che almeno due delle palline estratte portino lo stesso numero è

$$\mathbf{P}\left(A_q^C\right) = 1 - \mathbf{P}\left(A_q\right) = 1 - \prod_{i=1}^{q-1}\left(1 - \frac{i}{M}\right) = 1 - \prod_{k=2}^{q}\left(\frac{M-k+1}{M}\right).$$

Quindi, per rispondere al **problema dei compleanni**, si costruisce un modello nel quale i compleanni possibili sono 365, trascurando così la possibilità che un compleanno possa cadere il 29 febbraio; si eliminano cioè gli anni bisestili. Allora si ha $M = 365$. Con semplici calcoli si vede che anche il minimo numero q di presenti per il quale, nella notazione di sopra, è $\mathbf{P}(A_q^C) > \frac{1}{2}$ è dato da $q = 23$:

$$min\left\{q \in \mathbb{N} : \mathbf{P}\left(A_q^C\right) > \frac{1}{2}\right\} = 23.$$

Vale il seguente

Teorema 5.16. *L'**Algoritmo 5.15** è un (ε, q)-algoritmo, dove*

$$\varepsilon = 1 - \prod_{i=2}^{q}\left(\frac{M-i+1}{M}\right).$$

Dimostrazione. Siano $\mathcal{X}_0 = \{x_1, ..., x_q\}$ e $h(\mathcal{X}_0) = \{h(x_1), ..., h(x_q)\}$. Allora il problema delle collisioni in $\mathcal{X}_0$ è assimilabile al problema del compleanno. Infatti, basta assumere che x_i rappresenti una persona e che $h(x_i)$ rappresenti il suo compleanno. Pertanto, il problema di determinare una collisione è

$$\varepsilon = 1 - \prod_{i=2}^{q}\left(\frac{M-i+1}{M}\right).$$

□

Se M è elevato, allora $1-M \simeq e^{-M}$ e quindi la probabilià di non trovare collisioni è

$$\prod_{i=2}^{q}\left(1-\frac{i-1}{M}\right)=\prod_{k=1}^{q-1}\left(1-\frac{k}{M}\right)\simeq\prod_{k=1}^{q-1}e^{-\frac{k}{M}}=e^{-\Sigma_{k=1}^{q-1}\frac{k}{M}}=e^{-\frac{q(q-1)}{2M}}.$$

Quindi, la probabilità media di successo di trovare una collisione nella funzione hash nel modello dell'oracolo è $\varepsilon \simeq 1-e^{-\frac{q(q-1)}{2M}}$. Pertanto, $e^{-\frac{q(q-1)}{2M}} \simeq 1-\varepsilon$ implica $-\frac{q(q-1)}{2M} \simeq \ln(1-\varepsilon)$ da cui si ricava che $q^2-q \simeq 2M\ln\frac{1}{1-\varepsilon}$ e ignorando il termine $-q$ si ottiene

$$q \simeq \sqrt{2M\ln\frac{1}{1-\varepsilon}}.$$

Se $\varepsilon = 1/2$, allora $q \simeq 1.17\sqrt{M}$ pertanto, in tal caso l'**Algoritmo 5.15** è un $(1/2, O(\sqrt{M}))$.

- Se i sunti dei messaggi sono stringhe di 40 bit, cioè $M = 2^{40}$, la funzione hash non è sicura siccome una collisione potrebbe essere trovata con probabilità del 50% su 2^{20} (circa un milione) di hashes casuali.
- Ai fini della sicurezza la lunghezza minima richiesta dei sunti dei messaggi è di 128 bit, in pratica è raccomandato avere 160 bit.

5.6 Confronto tra i criteri di sicurezza

Dall'analisi degli **Algoritmi 5.11**, **5.13** e **5.15** si evince che, nel modello dell'oracolo il problema della collisione è più semplice da risolvere di quello della ricerca della seconda immagine inversa. Quest'ultimo, a sua volta, è più semplice da risolvere rispetto a quello dell'immagine inversa.

Proviamo che il problema della seconda immagine inversa e quello dell'immagine inversa possono essere ridotti al problema delle collisioni.

In particolare mostriamo che

- Una funzione hash $h : \mathcal{X} \longrightarrow \mathcal{Y}$ resistente alla collisione è anche resistenza alla ricerca della seconda immagine inversa.
- Una funzione hash $h : \mathcal{X} \longrightarrow \mathcal{Y}$, tale che $|\mathcal{X}| \geq 2|\mathcal{Y}|$ resistente alla collisione è anche resistenza alla ricerca della immagine inversa.

Sia $O2P$ un (ε, q)-algoritmo utilizzato per risolvere il problema della seconda immagine inversa.

Algoritmo 5.17. **(Collisione e seconda immagine inversa h)**

esterno $O2P$

Scelto $x \in \mathcal{X}$ in maniera casuale e uniforme

if $O2P(h, x) = x'$ **and** $(x \neq x')$ **and** $(h(x) = h(x'))$ **then return**(x, x')

else return (insuccesso).

Pertanto, se $O2P$ è (ε, q)-algoritmo, allora l'**Algoritmo 5.17** è $(\varepsilon, q+1)$. Quindi resistenza alla collisione implica resistenza alla ricerca dell'immagine inversa.

Vediamo ora la riduzione del problema della collisione a quello dell'immagine inversa. Sia ora $h : \mathcal{X} \longrightarrow \mathcal{Y}$ una funzione hash tale che $|\mathcal{X}| \geq 2|\mathcal{Y}|$ e OP un $(1, q)$-algoritmo utilizzato che risolve il problema dell'immagine inversa. Più precisamente, OP è un algoritmo che in corrispondenza dell'input $y \in \mathcal{Y}$ determina sempre almeno una $x \in \mathcal{X}$ (in particolare, h è suriettiva).

Algoritmo 5.18. **(Collisione e immagine inversa h)**

esterno OP

Scelto $x \in \mathcal{X}$ in maniera casuale e uniforme

$y \longleftarrow h(x)$

if$OP(h, y) = x'$ **and** $(x \neq x')$ **and then return**(x, x')

else return (insuccesso).

Vale il seguente

Teorema 5.19. *L'**Algoritmo 5.18** è un algoritmo Las Vegas di tipo* $\left(\frac{1}{2}, q+1\right)$.

Dimostrazione. Si ricordi che h è suriettiva. Quindi, l'insieme delle immagini inverse degli elementi di $\mathcal{Y}$ costituisce una partizione di $\mathcal{X}$ in $|\mathcal{Y}|$ classi.
Sia $x \in \mathcal{X}$ e denotata e si denoti con $[x]$ la classe data da $h^{-1}(h(x))$. Sia, infine $\{x_1, ..., x_{|\mathcal{Y}|}\}$ un sistema di rappresentanti delle classi.

Siccome OP un $(1,q)$-algoritmo, questo determina sicuramente un elemento di $[x]$. Quindi, fissato x in $\mathcal{X}$, la probabilità di ottenere una collisione è $\frac{|[x]|-1}{|[x]|}$. Pertanto, la probabilità media di successo nel determinare una collisione è

$$\begin{aligned}\mathbf{P} &= \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\frac{|[x]|-1}{|[x]|} = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{Y}|}\sum_{x\in[x_i]}\frac{|[x]|-1}{|[x]|}\\ &= \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{Y}|}\left(|[x_i]|\,\frac{|[x_i]|-1}{|[x_i]|}\right) = \frac{|\mathcal{X}|-|\mathcal{Y}|}{|\mathcal{X}|}\\ &\geq \frac{|\mathcal{X}|-|\mathcal{X}|/2}{|\mathcal{X}|} = 1/2.\end{aligned}$$

□

Pertanto, sotto le ipotesi fatte, la resistenza alla collisione implica la resistenza alla ricerca della immagine inversa.

5.7 Funzioni Hash Iterate

In questo capitolo studiamo le **funzioni hash iterate** che sono funzioni hash a dominio infinito ottenute estendendo funzioni di compressione che sono funzioni hash con dominio finito.

Gli input e gli output sono costituiti da stringhe binarie.

- $|x|$ denota la lunghezza della stringa binaria x (se $x = 10101$, allora $|x| = 5$).
- $x \parallel y$ denota la concatenazione delle stringhe x e y (se $x = 10$ e $y = 11$, allora $x \parallel y = 1011$).

Se compress : $\{0,1\}^{m+t} \longrightarrow \{0,1\}^m$, $m,t \geq 1$, è una qualsiasi funzione di compressione, la costruzione della funzione h consiste essenzialmente di tre passi:

1. **Elaborazione Preliminare** Data una stringa x tale che $|x| \geq m+t+1$, si costruisce una stringa

   $$y = y_1 \parallel y_2 \parallel \cdots \parallel y_r,$$

   in cui $|y_i| = t$ per $1 \leq t \leq r$.

   Generalmente, la costruzione di y è data $y = x \parallel \text{pad}(x)$, dove pad$(x)$ è una funzione che incorpora il valore binario di $|x|$ e bit addizionali (degli zero, per esempio) in modo tal che y abbia lunghezza un multiplo di t.

2. **Elaborazione** Se VI è una stringa binaria pubblica di lunghezza m (VI sta per **valore iniziale**) allora si precede come segue
$$\begin{array}{lll} z_0 & \longleftarrow & \text{VI} \\ z_1 & \longleftarrow & \text{compress}(z_0 \parallel y_1) \\ z_2 & \longleftarrow & \text{compress}(z_1 \parallel y_2) \\ \vdots & \vdots & \vdots \\ z_r & \longleftarrow & \text{compress}(z_{r-1} \parallel y_r). \end{array}$$

3. **Trasformazione opzionale di output** Sia $g : \{0,1\}^m \longrightarrow \{0,1\}^\ell$ una funzione pubblica, allora
$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \longrightarrow \{0,1\}^\ell, x \longmapsto g(z_r)$$
la funzione h è detta **funzione hash iterata**.

Remark 5.20. Si noti che l'applicazione $x \longmapsto y$ definita nell'Elaborazione preliminare deve essere iniettiva. Infatti se per distinti x, x' risulta $y = y'$, allora $h(x) = h(x')$ e quindi h non è resistente alla collisione. Pertanto, $|y| = rt \geq |x|$.

5.8 La costruzione di Merkle-Damgård

La costruzione di **Merkle-Damgård** (1990) fornisce una funzione hash iterata resistente alla collisione a partire da una funzione di compressione resistente alla collisione.

Figura 5.4: Ralph C. Merkle (1952) e Ivan Bjerre Damgård (1956)

Sia compress : $\{0,1\}^{m+t} \longrightarrow \{0,1\}^m$, $m, t \geq 1$, una qualsiasi funzione di compressione che gode della proprietà di essere resistente alla collisione e sia
$$\mathcal{X} = \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i.$$
Distinguiamo i due casi $t \geq 2$ e $t = 1$ che verranno trattati separatamente.

Supponiamo $t \geq 2$. Sia $n = |x| \geq m + t + 1$, allora

$$x = x_1 \parallel x_2 \parallel \cdots \parallel x_k$$

dove $|x_1| = |x_2| = \cdots = |x_{k-1}| = t - 1$ e $|x_k| = t - 1 - d$, dove $0 \leq d \leq t - 2$. Allora

$$n = (k-1)(t-1) + t - 1 - d = k(t-1) - d$$

quindi $k - 1 < k - \frac{d}{t-1} = \frac{n}{t-1} \leq k$ da cui segue

$$k = \left\lceil \frac{n}{t-1} \right\rceil.$$

allora $h(x)$ definito come l'output del seguente algoritmo.

Algoritmo 5.21. **(Merkle-Damgård**(x)**)**

esterno compress

commento: compress : $\{0,1\}^{m+t} \longrightarrow \{0,1\}^m$, $m, t \geq 1$

$n \longleftarrow |x|$

$k \longleftarrow \left\lceil \frac{n}{t-1} \right\rceil$

$d \longleftarrow k(t-1) - n$

for $i \longleftarrow 1$ **to** $k - 1$

do $y_i \longleftarrow x_i$

$y_k \longleftarrow x_k \parallel 0^d$

$y_{k+1} \longleftarrow$ la rappresentazione binaria di d

$z_1 \longleftarrow 0^{m+1} \parallel y_1$

$g_1 \longleftarrow \text{compress}(z_1)$

for $j \longleftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \longleftarrow g_i \parallel 1 \parallel y_{i+1} \\ g_{i+1} \longleftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \longleftarrow g_{k+1}$

return $(h(x))$

Sia $y(x) = y_1 \parallel y_2 \parallel \cdots \parallel y_k \parallel y_{k+1}$ dove $y_i = x_i$ per $i = 1, ..., k-1$ e $y_k = x_k \parallel 0^d$, allora $|y_i| = t - 1$ per $i = 1, ..., k - 1$. Infine, si possono aggiungere un opportuno numero di 0 alla sinistra alla rappresentazione binaria di d in modo da avere $|y_{k+1}| = t - 1$.

Remark 5.22.

- Per esempio, se $x = 1111100$ e $t = 4$, allora $n = 7$, $t-1 = 3$, $k = 3$ e $d = 2$. Chiaramente, $y_1 = x_1 = 111$, $y_2 = x_2 = 110$ e $x_3 = 0$ e $y_3 = 0 \parallel 0^2 = 000$. La rappresentazione binaria di d è 10 e quindi $y_4 = 0 \parallel 10 = 010$. Pertanto $y = 111110000010$.
- L'applicazione $x \longmapsto y(x)$ è iniettiva, e questo abbiamo visto essere condizione necessaria perché h sia resistente alla collisione.

Teorema 5.23. *Se la funzione* $\text{compress} : \{0,1\}^{m+t} \longrightarrow \{0,1\}^m$*, con* $t \geq 2$*, è resistente alla collisione, allora la funzione hash*

$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \longrightarrow \{0,1\}^m$$

*definita dall'**Algoritmo 5.21** è resistente alla collisione.*

Dimostrazione. Supponiamo che esistano distinti x, x' tali che $h(x) = h(x')$. Proviamo che troviamo una collisione nella funzione compress in tempo polinomiale. Siano

$$\begin{aligned} y(x) &= y_1 \parallel y_2 \parallel \cdots \parallel y_k \parallel y_{k+1} \\ y(x') &= y'_1 \parallel y'_2 \parallel \cdots \parallel y'_\ell \parallel y'_{\ell+1} \end{aligned}$$

dove x e x' sono stati allungati con d e d' zeri, rispettivamente. Siano $g_1, ..., g_{k+1}$ e $g'_1, ..., g'_{\ell+1}$ i corrispondenti valori di g nell'**Algoritmo 5.21**.
Distinguiamo i due casi a seconda $|x| \equiv |x'| \, mod \, t-1$ e $|x| \not\equiv |x'| \, mod \, t-1$.

Caso 1: $|x| \not\equiv |x'| \, mod \, t-1$.
Se $d = d'$, allora $k(t-1) - |x| = \ell'(t-1) - |x'|$ e quindi $|x| \equiv |x'| \, mod \, t-1$, assurdo. Pertanto, $d \neq d'$ e dalla definizione di y_{k+1} e $y'_{\ell+1}$, segue che $y_{k+1} \neq y'_{\ell+1}$. Allora

$$\begin{aligned} \text{compress}(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1} = h(x) = h(x') = \\ = g'_{\ell+1} = \text{compress}(g'_\ell \parallel 1 \parallel y'_{\ell+1}) \end{aligned}$$

è una collisione poiché $y_{k+1} \neq y'_{\ell+1}$.

Caso 2: $|x| \equiv |x'| \, mod \, t-1$.
Distinguiamo due sottocasi.

(a) $|x| = |x'|$.

$|x| = |x'|$ allora $k = \left\lceil \frac{|x|}{t-1} \right\rceil = \left\lceil \frac{|x'|}{t-1} \right\rceil = \ell$ e dalla definizione di y_{k+1} e y'_{k+1}, segue che $y_{k+1} = y'_{k+1}$. Allora

$$\begin{aligned} \text{compress}(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1} = h(x) = h(x') = \\ = g'_{k+1} = \text{compress}(g'_k \parallel 1 \parallel y_{k+1}). \end{aligned}$$

Se $g_k \neq g'_k$, abbiamo determinato una collisione in compress.

Se invece, $g_k = g'_k$, allora si ha

$$\text{compress}(g_{k-1} \parallel 1 \parallel y_k) = g_k = g'_k = \text{compress}(g'_{k-1} \parallel 1 \parallel y'_k).$$

e quindi o una collisione oppure $g_{k-1} = g'_{k-1}$ e $y_k = y'_k$.

Supponendo di non trovare collisioni dopo k passi, si ha

$$\text{compress}(0^{m+1} \parallel y_1) = g_1 = g'_1 = \text{compress}(0^{m+1} \parallel y'_1).$$

Se $y_1 \neq y'_1$ abbiamo determinato una collisione nella funzione compress, altrimenti $y(x) = y(x')$ e quindi $x = x'$ dall'initettvità di $x \longmapsto y(x)$, ma ciò è assurdo.

(b) $|x| \neq |x'|$.

Possiamo assumere che $|x'| > |x|$ e quindi $\ell > k$. Ragionando in modo analogo al caso precedente, o raggiungiamo una collisione negli ultimi k passi, oppure si giunge alla seguente situazione:

$$\text{compress}(0^{m+1} \parallel y_1) = g_1 = g'_{\ell-k+1} = \text{compress}(g'_{\ell-k} \parallel 1 \parallel y'_{\ell-k+1}).$$

Siccome l'$(m+1)$-esimo bit di $0^{m+1} \parallel y_1$ è 0, mentre l'$(m+1)$-esimo bit di $g'_{\ell-k} \parallel 1 \parallel y'_{\ell-k+1}$ è 1, abbiamo determinato una collisione.

□

La precedente costruzione era stata fatta nell'ipotesi che $t \geq 2$. Rimane da analizzare la costruzione di h per $t = 1$, la quale, come vedremo, avviene in modo differente.

Sia x una stringa binaria di lunghezza $|x| = n \geq m + 2$ e sia $f : \{0,1\} \to \{0,10\}$ tale che $f(0) = 0$ e $f(1) = 10$. La costruzione di h avviene attraverso il seguente algoritmo.

Algoritmo 5.24. **(Merkle-Damgård 2 (x))**

esterno compress
commento: compress : $\{0,1\}^{m+1} \longrightarrow \{0,1\}^m$, $m \geq 1$
$n \longleftarrow |x|$
$y \longleftarrow 11 \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$
sia $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, dove $y_i \in \{0,1\}$, $1 \leq i \leq k$
$g_1 \longleftarrow \text{compress}(0^m \parallel y_1)$
for $i \longleftarrow 1$ **to** $k-1$
do $g_{i+1} \longleftarrow \text{compress}(g_i \parallel y_{i+1})$
$h(x) \longleftarrow g_k$
return $(h(x))$

Osservazione 5.25. L'applicazione $x \longmapsto y(x)$ gode delle seguenti importanti proprietà:

- **Iniettività**. Supponiamo $x \neq x'$ e sia

$$I = \{1 \leq i \leq \min\{|x|, |x'|\} : x_i \neq x'_i\}.$$

  Se $I \neq \varnothing$, detto $i_0 = \min I$, segue che $f(x_j) = f(x'_j)$ per $1 \leq j \leq i_0 - 1$ e $f(x_{i_0}) \neq f(x'_{i_0})$, da cui segue che $y(x) \neq y(x')$.

  Se $I = \varnothing$, possiamo assumere $|x| < |x'|$ quindi $x' = w \parallel x$ e pertanto $y(x) \neq y(x')$.
- **Non esistono stringhe x, x', z con $x \neq x'$ tali che $y(x') = z \parallel y(x)$.**

  Supponiamo il contrario. Siano, quindi,

$$\begin{aligned} x &= x_1 \parallel x_2 \parallel \cdots \parallel x_n \\ x' &= x'_1 \parallel x'_2 \parallel \cdots \parallel x'_{n'} \end{aligned}$$

  tali che $x \neq x'$ e $y(x') = z \parallel y(x)$, dove z è un opportuna stringa binaria. Allora

$$11 \parallel f(x'_1) \parallel f(x'_2) \parallel \cdots \parallel f(x'_{n'}) = z \parallel 11 \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n).$$

Analizziamo il caso rimanente $t = 1$.

Teorema 5.26. *Se la funzione* $\text{compress} : \{0,1\}^{m+1} \longrightarrow \{0,1\}^m$ *è resistente alla collisione, allora la funzione hash*

$$h : \bigcup_{i=m+2}^{\infty} \{0,1\}^i \longrightarrow \{0,1\}^m$$

*definita dall'**Algoritmo 5.24** è resistente alla collisione.*

Dimostrazione. Supponiamo che esistano distinti x, x' tali che $h(x) = h(x')$. Proviamo che troviamo una collisione nella funzione compress in tempo polinomiale. Siano

$$\begin{aligned} y(x) &= y_1 \parallel y_2 \parallel \cdots \parallel y_k \parallel y_k \\ y(x') &= y'_1 \parallel y'_2 \parallel \cdots \parallel y'_\ell \parallel y'_\ell \end{aligned}$$

Siano $g_1, ..., g_k$ e $g'_1, ..., g'_\ell$ i corrispondenti valori nell'**Algoritmo 5.24**.

Distinguiamo i due casi a seconda $k \neq \ell$ o $k = \ell$.

Caso 1: $k = \ell$. Poiché
$$\text{compress}(g_{k-1} \parallel y_k) = g_k = h(x) = h(x') = \\ = g'_k = \text{compress}(g'_{k-1} \parallel y'_k),$$
se $g_{k-1} \neq g'_{k-1}$ o $y_k \neq y'_k$ abbiamo determinato una collisione in compress. Se, invece, $g_{k-1} = g'_{k-1}$ e $y_k = y'_k$ allora si ha
$$\text{compress}(g_{k-2} \parallel y_{k-1}) = g_k = g'_k = \text{compress}(g'_{k-2} \parallel y'_{k-1})$$
e quindi o una collisione oppure $g_{k-2} = g'_{k-2}$ e $y_{k-1} = y'_{k-1}$. Supponendo di non trovare collisioni dopo k passi segue che $y = y'$ e quindi $x = x'$ per l'iniettvità di $x \longmapsto y(x)$. Ma questo è un assurdo poichè $x \neq x'$ per ipotesi.

Caso 2: $k \neq \ell$. Possiamo supporre che $\ell > k$. Poiché
$$\text{compress}(g_{k-1} \parallel y_k) = g_k = h(x) = h(x') = g'_\ell = \text{compress}(g'_{\ell-1} \parallel y'_\ell)$$
e $g_{k-1} \neq g'_{\ell-1}$ o $y_k \neq y'_\ell$ abbiamo determinato una collisione in compress. Se, invece, $g_{k-1} = g'_{\ell-1}$ e $y_k = y'_\ell$ allora si ha
$$\text{compress}(g_{k-2} \parallel y_{k-1}) = g_k = g'_\ell = \text{compress}(g'_{\ell-2} \parallel y'_{\ell-1})$$
e quindi o una collisione oppure $g_{k-2} = g'_{\ell-2}$ e $y_{k-1} = y'_{\ell-1}$. Supponendo di non trovare collisioni, dopo k passi segue che $y_{k-j} = y'_{\ell-j}$ per $j = 0, ..., k-1$. Quindi, $y(x') = z \parallel y(x)$ dove $z = y'_1 \parallel y'_2 \parallel \cdots \parallel y'_{\ell-k}$, ma ciò contraddice la seconda proprietà dell'applicazione $x \longmapsto y(x)$ (vedi **Osservazione 5.25**).

□

Il seguente teorema sintetizza le costruzioni di Merkle-Damgård di funzioni hash iterate appena viste.

Teorema 5.27. *Se la funzione* compress : $\{0,1\}^{m+t} \longrightarrow \{0,1\}^m$, *con* $t \geq 1$, *è resistente alla collisione, allora esiste una funzione hash*
$$h : \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \longrightarrow \{0,1\}^m$$
resistente alla collisione.
Se l'input è una stringa binaria di lunghezza n, *il numero di volte che la funzione* compress *è calcolata per determinare* h *è al più*
$$\begin{array}{ll} 1 + \left\lceil \frac{n}{t-1} \right\rceil & \text{se } t \geq 2 \\ 2n + 2 & \text{se } t = 1. \end{array}$$

5.9 SHA-1

In questa sezione descriviamo una funzione hash iterata nota come SHA-1 che è l'acronimo di **The Secure Hash Algorithm** (1995).

Caratteristiche generali:

- Input è una stringa binaria x tale che $|x| \leq 2^{64} - 1$, quindi la rappresentazione binaria di $|x|$ ha lunghezza al più 64 bit.
- Attraverso il processo di imbottitura estende x ad una stringa binaria di lunghezza un multiplo di 512.
- La funzione di compressione mappa $160 + 512$ bit in 160 bit che rappresentano l'output.
- Le operazioni che sono alla base del funzionamento di SHA-1 avvengono su **word** che sono stringhe binare di lunghezza 32 (o equivalentemente stringhe esadecimali di lunghezza 8) e sono le seguenti

Siano $X = x_0 \cdots x_{31}$ e $Y = y_0 \cdots y_{31}$ due parole, allora

$X \wedge Y = z_0 \cdots z_{31}$	$z_i = 1 \iff x_i = y_i = 1$
$X \vee Y = z_0 \cdots z_{31}$	$z_i = 1 \iff x_i = 1$ or $y_i = 1$
$X \oplus Y = z_0 \cdots z_{31}$	$z_i = (x_i + y_i) mod\ 2$
$\neg X = X \oplus 1^{32}$	
$X + Y = z_0 \cdots z_{31}$	$\sum_{i=0}^{31} z_i 2^i = \left(\sum_{i=0}^{31} x_i 2^i + \sum_{i=0}^{31} y_i 2^i\right) mod\ 2^{32}$
$ROT^s(X) = x_{31-s+1} \cdots x_1 \cdots x_{31-s} \quad 0 \leq s \leq 31$	

La prima fase detta **Imbottitura** è descritta nel seguente algoritmo

Algoritmo 5.28. **(SHA-1-PAD(x))**

comment: $|x| \leq 2^{64} - 1$
$d \longleftarrow (447 - |x|) mod\ 512$
$\ell \longleftarrow 0^{64-\text{ rappresentazione binaria di } |x|} \parallel$ rappresentazione binaria di $|x|$.
$y \longleftarrow x \parallel 1 \parallel 0^d \parallel \ell$.

Si noti che nella fase di imbottitura si concatena ad x il $1 \parallel 0^d$ e la rappresentazione binaria di $|x|$ ottenendo così un stringa $|x| + 1 + (447 - |x|) \ mod\ 512 + 64$ che è un multiplo di 512.

Esempio 5.29. Se $x = 2^{32}$, quindi in binario x è $1 \parallel 0^{32}$ e quindi $|x| = 33$ e $d = 414$. Siccome $|x| = 2^5 + 1$ in binario è 100001, allora $\ell = 0^{58} \parallel 100001$. Quindi,

$$y = x \parallel 1 \parallel 0^d \parallel \ell = \left(1 \parallel 0^{32}\right) \parallel 1 \parallel 0^{414} \parallel 0^{58} \parallel 100001.$$

La stringa risultante dell'imbottitura viene suddivisa in n blocchi di 512 bit ognuno:

$$y = M_1 \parallel M_2 \parallel \cdots \parallel M_n.$$

Definiamo le seguenti funzioni $f_0, ..., f_{79}$ come segue:

$$f_i(B,C,D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & \text{se} \quad 0 \leq t \leq 19 \\ B \oplus C \oplus D & \text{se } 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{se } 40 \leq t \leq 59 \\ B \oplus C \oplus D & \text{se } 60 \leq t \leq 79 \end{cases}$$

Ogni funzione f_i associa ad ogni 3 word 1 word.
Definiamo le word costanti $K_0, ..., K_{79}$ (in notazione esadecimale)

$$K_i = \begin{cases} \texttt{5A827999} & \text{se} \quad 0 \le t \le 19 \\ \texttt{6ED9EBA1} & \text{se } 20 \le t \le 39 \\ \texttt{8F1BBCDC} & \text{se } 40 \le t \le 59 \\ \texttt{CA62C1D6} & \text{se } 60 \le t \le 79 \end{cases}$$

Vediamo ora come funziona la SHA-1

Algoritmo 5.30. (SHA-1(x))

external: *SHA-1-PAD*
global: $K_0, ..., K_{79}$
$y \longleftarrow SHA\text{-}1\text{-}PAD(x)$
denote $y = M_1 \parallel M_2 \parallel \cdots \parallel M_n$ dove ogni M_i è un blocco di 512 bit
$H_0 \longleftarrow \texttt{67452301}$
$H_1 \longleftarrow \texttt{EFCDAB89}$
$H_2 \longleftarrow \texttt{98BADCFE}$
$H_3 \longleftarrow \texttt{10325476}$
$H_4 \longleftarrow \texttt{C3D2E1F0}$
for $i \longleftarrow 1$ **to** n
do
- denote $M_i = W_0 \parallel W_1 \parallel \cdots \parallel W_{15}$, dove W_i è una word
- **for** $t \longleftarrow 16$ **to** 79
- **do** $W_t \longleftarrow ROT^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$
- $A \longleftarrow H_0$
- $B \longleftarrow H_1$
- $C \longleftarrow H_2$
- $D \longleftarrow H_3$
- $E \longleftarrow H_4$
- **for** $t \longleftarrow 0$ **to** 79
- **do**
  - $temp \longleftarrow ROT^5(A) + f_t(B,C,D) + E + W_t + K_t$
  - $E \longleftarrow D$
  - $D \longleftarrow C$
  - $C \longleftarrow ROT^{30}(B)$
  - $B \longleftarrow A$
  - $A \longleftarrow temp$
- $H_0 \longleftarrow H_0 + A$
- $H_1 \longleftarrow H_1 + B$
- $H_2 \longleftarrow H_2 + C$
- $H_3 \longleftarrow H_3 + D$
- $H_4 \longleftarrow H_4 + E$

return $(H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4)$.

5.10 Message Authentication Code (MAC)

I requisiti di sicurezza di un **Message Authentication Code**, brevemente **MAC**, sono descritti qui di seguito.

Definizione 5.31. **((q,ε)-falsificazione esistenziale)**
Sia $(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$ un MAC e siano $(x_1,y_1),\cdots,(x_q,y_q)$ coppie valide sotto una chiave fissata K, i.e. $y_i = h_K(x_i)$ per ogni $i = 1,\cdots,q$, richieste da un avversario all'oracolo. Se (x,y) è una coppia creata dall'avversario tale che $x \notin \{x_1,\cdots,x_q\}$ e $\boldsymbol{P}[y = h_K(x)] \geq \varepsilon$, allora (x,y) è una **(q,ε)-falsificazione esistenziale**.

Un MAC viene considerato **sicuro** se lo è rispetto alle (q,ε)-falsificazioni esistenziali.

Un modo comune di costruire un MAC è quello di considerare una funzione hash h priva di chiave e di inserire una chiave segreta nel messaggio da cifrare.

Esempio 5.32. Supponiamo che $h : \mathcal{X} \longrightarrow \mathcal{Y}$ sia una funzione hash iterata che non abbia sia l'elaborazione preliminare che la trasformazione opzionale di output.
Siano $\text{compress} : \{0,1\}^{m+t} \longrightarrow \{0,1\}^m$, $m,t \geq 1$, una qualsiasi funzione di compressione (pubblica) e $\text{VI} = K$ il valore iniziale che è una chiave segreta costituita da m-bit e supponiamo che la lunghezza del generico messaggio da cifrare sia sempre un multiplo di t.
Quindi, se $x = x_1 \parallel x_2 \parallel \cdots \parallel x_r$ e $|x_i| = t$ per ogni $i = 1,...,r$, allora h_K è definita come segue:

$$\begin{array}{lcl} z_0 & \longleftarrow & \text{VI} \\ z_1 & \longleftarrow & \text{compress}(z_0 \parallel x_1) \\ z_2 & \longleftarrow & \text{compress}(z_1 \parallel x_2) \\ \vdots & \vdots & \vdots \\ z_r & \longleftarrow & \text{compress}(z_{r-1} \parallel x_r). \\ h_K(x) & \longleftarrow & z_r \end{array}$$

Un potenziale avversario che conosce la coppia valida $(x,h_K(x))$ nel modello dell'oracolo, produce coppie valide $(\bar{x},h_K(\bar{x}))$, senza conoscere K, come segue:

1. sceglie un qualsiasi stringa binaria x' di lunghezza t e considera il messaggio $x \parallel x'$
2. Calcola $h_K(x \parallel x') = \text{compress}(h_K(x) \parallel x')$.

Esempio 5.33. Supponiamo che $h : \mathcal{X} \longrightarrow \mathcal{Y}$ sia una funzione hash iterata che abbia sia l'elaborazione preliminare ma non la trasformazione opzionale di output.

Siano compress e VI definiti come sopra. Nella fase di elaborazione preliminare sia $y = x \parallel \text{pad}(x)$ tale che $y = y_1 \parallel y_2 \parallel \cdots \parallel y_r$ e $|y_i| = t$ per ogni $i = 1, ..., r$, allora h_K è definita come segue:

$$\begin{array}{lll} z_0 & \longleftarrow & \text{VI} \\ z_1 & \longleftarrow & \text{compress}(z_0 \parallel y_1) \\ z_2 & \longleftarrow & \text{compress}(z_1 \parallel y_2) \\ \vdots & \vdots & \vdots \\ z_r & \longleftarrow & \text{compress}(z_{r-1} \parallel y_r). \\ h_K(x) & \longleftarrow & z_r \end{array}$$

Un potenziale avversario che conosce la coppia valida $(x, h_K(x))$ nel modello dell'oracolo, può produrre coppie valide $(x', h_K(x'))$, senza conoscere K, come segue:

1. Sceglie una qualsiasi stringa binaria w di lunghezza t e considera il messaggio $x' = x \parallel \text{pad}(x) \parallel w$.
2. Calcola $y' = x' \parallel \text{pad}(x') = x \parallel \text{pad}(x) \parallel w \parallel \text{pad}(x')$.

   Quindi, $y' = y_1 \parallel \cdots \parallel y_r \parallel y_{r+1} \parallel \cdots \parallel y_{r'}$ con $|y_i| = t$.
3. Calcola $h_K(x')$ come segue

$$\begin{array}{lll} z_{r+1} & \longleftarrow & \text{compress}(h_K(x) \parallel y_{r+1}) \\ z_{r+2} & \longleftarrow & \text{compress}(z_{r+1} \parallel y_{r+2}) \\ \vdots & \vdots & \vdots \\ z_{r'} & \longleftarrow & \text{compress}(z_{r'-1} \parallel y_{r'}). \\ h_K(x') & \longleftarrow & z_{r'} \end{array}$$

La coppia $(x', h_K(x'))$ è chiaramente valida.

Si noti che negli esempi precedenti l'avversario ha prodotto $(1, 1)$-falsificazioni.

5.11 Nested MAC

Definizione 5.34. **(Nested MAC)**
Siano $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{G})$ e $(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$ due famiglie hash tali che $|\mathcal{X}| > |\mathcal{Y}| \geq |\mathcal{Z}|$, allora un **Nested MAC** è una famiglia hash $(\mathcal{X}, \mathcal{Z}, \mathcal{M}, \mathcal{G} \circ \mathcal{H})$ in cui

1. $\mathcal{M} = \mathcal{L} \times \mathcal{K}$.
2. $\mathcal{H} \circ \mathcal{G} = \{h \circ g : h \in \mathcal{H}, g \in \mathcal{G}\}$.
3. Per ogni $(L, K) \in \mathcal{M}$ vale che $(h \circ g)_{(L,K)}(x) = h_L(g_K(x))$ per ogni $x \in \mathcal{X}$.

Si noti che $(\mathcal{X}, \mathcal{Z}, \mathcal{M}, \mathcal{G} \circ \mathcal{H})$ e $(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$ sono detti **Big MAC** e **Little MAC**, rispettivamente.

Proviamo che se valgono

(i) per ogni chiave (segreta) $K \in \mathcal{K}$ la famiglia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{G})$ è resistente alla collisione

(ii) per ogni chiave (segreta) $L \in \mathcal{L}$ il MAC $(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$ è sicuro

allora il Nested MAC è sicuro rispetto ai seguenti tipi di attacchi (**Teorema 5.35**):

I. Big MAC attack: In questo attacco una chiave (L, K) è scelta e tenuta segreta. Un avversario sceglie $x_1, x_2, ..., x_q$ messaggi e ha la possibilità per ogni x_i di sapere il valore di $h_L(g_K(x_i))$ nel modello dell'oracolo. Quindi, l'avversario tenta di generare una coppia valida $(x', h_L(g_K(x')))$.

II. Little MAC attack: In questo attacco una chiave L è scelta e tenuta segreta. Un avversario sceglie $y_1, y_2, ..., y_q$ messaggi e ha la possibilità per ogni y_i di sapere il valore di $h_L(y_i)$ nel modello dell'oracolo. Quindi, l'avversario tenta di generare una coppia valida $(y', h_L(y'))$.

III. Unknown-key collision attack: In questo attacco una chiave K è scelta e tenuta segreta. Un avversario sceglie $x_1, x_2, ..., x_q$, x messaggi distinti e ha la possibilità per ogni x_i di sapere il valore di $y_i = g_K(x_i)$, così come $y = g_K(x)$, nel modello dell'oracolo. Se $y = y_{i_0}$, allora l'avversario ha ottenuto una collisione. Quindi per ogni chiave segreta L fissata vale che $(x, h_L(y))$ è una coppia valida per il Little MAC $(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$.

Teorema 5.35. *Sia $(\mathcal{X}, \mathcal{Z}, \mathcal{M}, \mathcal{G} \circ \mathcal{H})$ un nested MAC, costruito dai MAC $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{G})$ e $(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$, tale che gode delle seguenti proprietà:*

1. *Per ogni $K \in \mathcal{K}$, K segreta, e per ogni $x_1, ..., x_{q+1}$ scelti dall'avversario vale che*
$$\boldsymbol{P}[g_K(x_i) = g_K(x_j), 1 \le i, j \le q+1, i \ne j] < \varepsilon_1$$
2. *Per ogni $L \in \mathcal{L}$, L segreta, e per ogni $(y_1, h_L(y_1)), ...(y_q, h_L(y_q))$ chiesti all'oracolo vale che*
$$\boldsymbol{P}[(y, h_L(y)), y \ne y_i, 1 \le i \le q] < \varepsilon_2.$$

Se esiste una (q, ε)-falsificazione per il nested MAC per una funzione $(h \circ g)_{(L,K)} \in \mathcal{H} \circ \mathcal{G}$ scelta in modo casuale, allora $\varepsilon < \varepsilon_1 + \varepsilon_2$.

Dimostrazione. Siano $x_1, \cdots, x_q$ i messaggi scelti dall'avversario e siano $z_1 = (h \circ g)_{(L,K)}(x_1), \cdots, z_q = (h \circ g)_{(L,K)}(x_1)$ i corrispondenti valori ottenuti chiedendo all'oracolo.

- Sia A l'evento (q, ε)-falsificazione, cioè l'evento (x, z), con $z = (h \circ g)_{(L,K)}(x)$ e $x \notin \{x_1, \cdots, x_q\}$. Allora $\boldsymbol{P}(A) \ge \varepsilon$.
- Sia B l'evento $y \in \{y_1, \cdots, y_q\}$, dove $y = g_K(x)$ e $y_i = g_K(x_i)$, allora per **(1)** vale che $\boldsymbol{P}(B) < \varepsilon_1$.

Quindi

$$\varepsilon \leq \boldsymbol{P}(A) = \boldsymbol{P}(A \cap B) + \boldsymbol{P}(A \cap B^c) \leq \boldsymbol{P}(B) + \boldsymbol{P}(B^c) < \varepsilon_1 + \boldsymbol{P}(B^c)$$

Pertanto,

$$\varepsilon - \varepsilon_1 < \boldsymbol{P}(B^c).$$

La probabilità dell'evento $B^c : y \notin \{y_1, ..., y_q\}$ è uguale alla probabilità che (y, z) sia una falsificazione. Siccome questa è minore di ε_2 **(2)**, segue che $\boldsymbol{P}(B^c) < \varepsilon_2$. Quindi, $\varepsilon - \varepsilon_1 < \varepsilon_2$, ovvero

$$\varepsilon < \varepsilon_1 + \varepsilon_2.$$

□

5.11.1 HMAC

Un **HMAC** è un nested MAC proposto da FIPS ed è costruito a partire da una funzione hash priva di chiave. Noi vedremo una versione basata sulla funzione SHA-1. Siano

1. K una chiave segreta rappresentata da una stringa binaria di 512 bit
2. *ipad* e *opad* delle stringhe binarie costanti ognuna di 512 bit che per semplicità sono rappresentate in esadecimale

$$\begin{aligned} ipad &= \texttt{3636}\cdots\texttt{36} \\ opad &= \texttt{5C5C}\cdots\texttt{5C} \end{aligned}$$

3. HMAC definito come segue:

$$\text{HMAC}_K(x) = \text{SHA-1}(K \oplus opad \parallel \text{SHA-1}(K \oplus ipad \parallel x)).$$

Si noti che

$$g_{K \oplus ipad}(x) = \text{SHA-1}(K \oplus opad \parallel x)$$

e

$$h_{K \oplus opad}(y) = \text{SHA-1}(K \oplus opad \parallel y).$$

Assumendo che $g_{K \oplus ipad}$ sia resistente alla collisione e $h_{K \oplus opad}$ sia sicura come MAC, segue dal **Teorema 5.35** che $\text{HMAC}_K(x)$ è sicuro come MAC.

5.11.2 CBC – MAC

Il modo più comune di costruire MAC è quello di utilizzare un cifrario a blocchi nella modalità CBC con un fissato, pubblico vettore di inizializzazione.

Sia $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un sistema crittografico in cui $\mathcal{P} = \mathcal{C} = \{0,1\}^t$. Sia VI il vettore nullo di lunghezza t, K una chiave segreta e il messaggio $x = x_1 \parallel x_2 \parallel \cdots \parallel x_n$ e $|x_i| = t$ per ogni $i = 1, ..., n$ allora **CBC – MAC** funziona come segue

Algoritmo 5.36. **(CBC – MAC(x, K))**

Sia $x = x_1 \| x_2 \| \cdots \| x_n$
VI $\longleftarrow$ 00···0
$y_0 \longleftarrow$ VI
for $i \longleftarrow 1$ **to** n
  do $y_i \longleftarrow e_K(y_{i-1} \oplus x_i)$
return (y_n)

Si noti che $h_K(x) = y_n$.

Il più noto attacco al CBC – MAC è l'attacco basato sul paradosso dei compleanni (sulle collisioni):

Siano $n \geq 2$ e $x_3, ..., x_n$ stringhe binarie di lunghezza t. Sia $q \simeq 1.17 \times 2^{t/2}$ un intero e si scelgano q **distinte** stringhe binarie $x_1^1, ..., x_1^q$ ognuna di lunghezza t. Siano q stringhe binarie $x_2^1, ..., x_2^q$ ognuna di lunghezza t scelte in maniera casuale. Per ogni $i = 1, ..., q$ si definiscano le stringhe

$$x^i = x_1^i \| x_2^i \| x_3 \| \cdots \| x_n$$

Si noti che per ogni $i \geq 1$ e per ogni $e \geq 3$ vale che $x_e^i = x_e$. Inoltre, $x^i \neq x^j$ per $i \neq j$ siccome $x_1^i \neq x_1^j$.

A questo punto l'avversario chiede all'oracolo il MAC di ogni x^i, cioè $h_K(x^i)$.

Proposizione 5.37. Vale che

$$h_K(x^i) = h_K(x^j) \iff y_1^i \oplus x_2^i = y_1^j \oplus x_2^j.$$

Dimostrazione.

($\Leftarrow$) Se $y_1^i \oplus x_2^i = y_1^j \oplus x_2^j$, allora $y_2^i = e_K\left(y_1^i \oplus x_2^i\right) = e_K(y_1^j \oplus x_2^j) = y_2^j$. Inoltre, siccome $x_e^i = x_e = x_e^j$ per $e \geq 3$ vale che $e_K(y_{e-1}^i \oplus x_e) = e_K(y_{e-1}^j \oplus x_e)$ e quindi $h_K(x^i) = h_K(x^j)$.

($\Rightarrow$) Viceversa, se $h_K(x^i) = h_K(x^j)$, allora $e_K(y_{n-1}^i \oplus x_n) = e_K(y_{n-1}^j \oplus x_n)$. Siccome $\mathcal{P} = \mathcal{C} = \{0,1\}^t$ allora e_K è bigettiva e quindi $y_{n-1}^i \oplus x_n = y_{n-1}^j \oplus x_n$ da cui segue immediatamente attraverso una somma XOR che $y_{n-1}^i = y_{n-1}^j$. Ripetendo questo procedimento $n-2$ volte si ottiene $y_1^i \oplus x_2^i = y_1^j \oplus x_2^j$.

□

Poiché $q \simeq 1.17 \times 2^{t/2}$ segue dal paradosso dei compleanni che, con probabilià di almeno $1/2$ esistono $1 \leq i_0, j_0 \leq q$, con $i_0 \neq j_0$, tali che $h_K(x^{i_0}) = h_K(x^{j_0})$, quindi per la **Proposizione 5.37**, $y_1^{i_0} \oplus x_2^{i_0} = y_1^{j_0} \oplus x_2^{j_0}$. Sia x_δ una qualsiasi stringa binaria di lunghezza t e si definiscano

$$\begin{aligned} v &= x_1^{i_0} \parallel \left(x_2^{i_0} \oplus x_\delta\right) \parallel x_3 \parallel \cdots \parallel x_n \\ w &= x_1^{j_0} \parallel \left(x_2^{j_0} \oplus x_\delta\right) \parallel x_3 \parallel \cdots \parallel x_n \end{aligned}$$

L'avversario chiede il MAC di v, quindi determina la coppia valida $(v, h_K(v))$. Siccome $y_1^{i_0} \oplus x_2^{i_0} \oplus x_\delta = y_1^{j_0} \oplus x_2^{j_0} \oplus x_\delta$ vale che $h_K(v) = h_K(w)$. Pertanto, l'avversario ha prodotto la coppia valida $(w, h_K(w))$ ovvero esso ha generato una $(O(2^{t/2}), 1/2)$-falsificazione.

5.12 MAC incondizionatamente sicuri

In questa sezione definiamo le funzioni hash fortemente universali e studiamo la loro applicazione nella costruzione di **MAC incondizionatamente sicuri**.

Supponiamo che ogni chiave sia usata solo per un'unica autenticazione. Pertanto, un avversario può fare al più una richiesta di una coppia valida prima che esso produca una possibile falsificazione. Ovvero, costruiamo dei MAC in cui non esistono (q, ε)-falsificazioni, dove $q = 0, 1$ e ε è opportuno, nemmeno se l'avversario ha a disposizione infinite capacità computazionali.

Supponiamo che due utenti A e B abbiano scelto una chiave segreta K_0 per autenticare un unico messaggio

- $(0, \varepsilon)$-falsificazione (**imitazione**).

  Per ogni $x \in \mathcal{X}$ e $y \in \mathcal{Y}$ la probabilità che (x, y) sia una coppia valida rispetto alla chiave K_0 è detto Ricavo(x, y). Se la distribuzione di probabilià delle chiavi è uniforme, allora

  $$\text{Ricavo}(x, y) = \mathbf{P}[y = h_{K_0}(x)] = \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|}.$$

  Quindi, un avversario nella produzione di una falsificazione deve scegliere una coppia (x, y) per cui il Ricavo(x, y) è il più grande possibile. Questa prende il nome di **Probabilità di Inganno**:

  $$Pd_0 = \max\{\text{Ricavo}(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}\}.$$

- $(1, \varepsilon)$-falsificazione (**sostituzione**).

  Supponiamo che (x, y) sia una coppia valida ottenuta da un avversario mediante richiesta all'oracolo. Sia $x' \in \mathcal{X}$, $x' \neq x$ e $y' \in \mathcal{Y}$ allora la probabilità che (x', y') sia una coppia valida rispetto alla chiave K_0, sapendo che (x, y) lo è rispetto a K_0, si dice Ricavo$(x', y'; x, y)$.

Quindi,

$$\text{Ricavo}(x', y'; x, y) = \mathbf{P}[y' = h_{K_0}(x') \mid y = h_{K_0}(x)] = \frac{\mathbf{P}[y' = h_{K_0}(x'), y = h_{K_0}(x)]}{\mathbf{P}[y = h_{K_0}(x)]} =$$
$$= \frac{|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|}{|\mathcal{K}|} \times \frac{|\mathcal{K}|}{|\{K \in \mathcal{K} : h_K(x) = y\}|} =$$
$$= \frac{|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|}{|\{K \in \mathcal{K} : h_K(x) = y\}|}.$$

Quindi, data una coppia valida (x, y), un avversario nella produzione di una falsificazione deve scegliere una coppia (x', y') per cui il $\text{Ricavo}(x', y'; x, y)$ è il più grande possibile.

Sia $\mathcal{V}$ l'insieme delle coppie che sono valide per almeno una chiave

$$\mathcal{V} = \{(x, y) : |\{K \in \mathcal{K} : h_K(x) = y\}| \geq 1\},$$

allora la **Probabilità di Inganno** è

$$Pd_1 = \max\{\text{Ricavo}(x', y'; x, y) : x, x' \in \mathcal{X}, y, y' \in \mathcal{Y}, (x, y) \in \mathcal{V}, x' \neq x\}.$$

Nella costruzione di un MAC in cui una chiave è utilizzata solo per un'autenticazione deve risultare che le probabilità di inganno Pd_0, Pd_1 devono essere uniformi e minimizzate.

Esempio 5.38. Si consideri il MAC $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ in cui $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_3$, $\mathcal{K} = \mathbb{Z}_3^2$ e $\mathcal{H} = \{h_{(a,b)} : (a, b) \in \mathbb{Z}_3^2\}$, dove $h_{(a,b)}(x) = ax + b \ mod\ 3$. Vogliamo determinarne le probabilità di inganno Pd_0, Pd_1.

Si consideri la **Matrice di Autenticazione** della famiglia hash $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ in cui le righe sono indicizzate dagli elementi di $\mathcal{K}$, le colonne da quelli di $\mathcal{X}$. Quindi il termine di posto (a, b) e x è $h_{(a,b)}(x)$. Così

$\mathcal{K}$ \ $\mathcal{X}$	**0**	**1**	**2**
(0,0)	0	0	0
(0,1)	1	1	1
(0,2)	2	2	2
(1,0)	0	1	2
(1,1)	1	2	0
(1,2)	2	0	1
(2,0)	0	2	1
(2,1)	1	0	2
(2,2)	2	1	0

Tabella 8: Matrice d'autenticazione.

Si noti che ognuno dei simboli $0, 1, 2$ compare esattamente 3 volte su ogni colonna. Pertanto, per ogni coppia (x, y) vale che $\text{Ricavo}(x, y) = 3/9 = 1/3$. Quindi $Pd_0 = 1/3$.

Si noti che ogni coppia valida individua esattamente 3 righe nella matrice di autenticazione ed è facile vedere che due qualsiasi coppie valide hanno esattamente una chiave a comune. Pertanto, fissata una coppia valida (x, y) risulta che per ogni altra coppia (x', y') si ha Ricavo$(x', y'; x, y) = 1/3$. Quindi, $Pd_1 = 1/3$.

5.12.1 Famiglie hash fortemente universali

Definizione 5.39. **(Famiglia hash fortemente universale)**
Una (N, M) - famiglia hash $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ si dice **fortemente universale** se, e solo se, per ogni $x, x' \in \mathcal{X}$, $x \neq x'$, e per ogni $y, y' \in \mathcal{Y}$ vale che

$$|\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}| = \frac{|\mathcal{K}|}{M^2}$$

La $(3, 3)$-famiglia hash dell'**Esempio 5.38** è **fortemente universale**.

Lemma 5.40. *Se $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è una (N, M)-famiglia hash fortemente universale, allora per ogni ogni $x \in \mathcal{X}$ e $y \in \mathcal{Y}$*

$$|\{K \in \mathcal{K} : h_K(x) = y\}| = \frac{|\mathcal{K}|}{M}.$$

Dimostrazione. Fissati $x, x' \in \mathcal{X}$ con $x' \neq x$ e $y \in \mathcal{Y}$, risulta che

$$\{K \in \mathcal{K} : h_K(x) = y\} = \dot{\bigcup_{y' \in \mathcal{Y}}} \{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}.$$

Pertanto,

$$\begin{aligned} |\{K \in \mathcal{K} : h_K(x) = y\}| &= \sum_{y' \in \mathcal{Y}} |\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}| \\ &= \sum_{y' \in \mathcal{Y}} \frac{|\mathcal{K}|}{M^2} = M \frac{|\mathcal{K}|}{M^2} = \frac{|\mathcal{K}|}{M}, \end{aligned}$$

essendo $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ fortemente universale.

□

Teorema 5.41. *Ogni (N, M)-famiglia hash fortemente universale è un MAC con $Pd_0 = Pd_1 = 1/M$*

Dimostrazione. Sia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ una (N, M)-famiglia hash fortemente universale. Allora, segue dal **Lemma 5.40** e dalla definizione di Ricavo che

$$\text{Ricavo}(x, y) = \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|} = \frac{1}{|\mathcal{K}|}\frac{|\mathcal{K}|}{M} = \frac{1}{M}.$$

Pertanto, $Pd_0 = 1/M$.
Siano $x, x' \in \mathcal{X}$ e $y, y' \in \mathcal{Y}$ tali che $x \neq x'$ e $(x, y) \in \mathcal{V}$, allora

$$\text{Ricavo}(x', y'; x, y) = \frac{|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|}{|\{K \in \mathcal{K} : h_K(x) = y\}|} = \\ = \frac{|\mathcal{K}|/M^2}{|\mathcal{K}|/M} = \frac{1}{M}$$

siccome $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è fortemente universale e siccome vale il **Lemma 5.40**. Quindi, $Pd_1 = 1/M$.

□

Forniamo degli esempi di famiglie hash fortemente universali.

Esempio 5.42. Sia p un primo e sia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ in cui $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p$, $\mathcal{K} = \mathbb{Z}_p^2$ e $\mathcal{H} = \left\{h_{(a,b)} : (a, b) \in \mathbb{Z}_p^2\right\}$, dove $h_{(a,b)}(x) = ax + b \ mod\ p$. Allora $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è una (p, p)-famiglia fortemente universale.

Dimostrazione. Siano $x, x', y, y' \in \mathbb{Z}_p$ con $x \neq x'$ allora esiste un unica chiave $(a, b) \in \mathbb{Z}_p^2$ tale che

$$\begin{cases} ax + b = y \ mod\ p \\ ax' + b = y' \ mod\ p \end{cases}$$

Siccome $x \neq x'$ in $\mathbb{Z}_p$ allora esiste $(x' - x')^{-1}$ in $\mathbb{Z}_p$ e quindi

$$\begin{cases} a = \frac{y'-y}{x'-x} \ mod\ p \\ b = y - x\frac{y'-y}{x'-x} \ mod\ p \end{cases}$$

sono univocamente determinati. Pertanto

$$\left|\left\{(a, b) \in \mathcal{K} : h_{(a,b)}(x) = y, h_{(a,b)}(x') = y'\right\}\right| = 1 = \frac{|\mathcal{K}|}{M^2}$$

e quindi $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è una (p, p)-famiglia fortemente universale.

□

Esempio 5.43. Siano ℓ un intero positivo, p un primo e $\mathcal{X} = \{0,1\}^\ell - \{(0,...,0)\}$. Per ogni $\vec{r} \in \mathbb{Z}_p^\ell$ sia $f_{\vec{r}} : \mathcal{X} \longrightarrow \mathbb{Z}_p, \vec{x} \longmapsto (\vec{r} \cdot \vec{x}) \mod p$, dove $\vec{r} \cdot \vec{x} = \sum_{i=1}^{\ell} r_i x_i$. Allora $\left(\mathcal{X}, \mathbb{Z}_p, \mathbb{Z}_p^\ell, \left\{f_{\vec{r}} : \vec{r} \in \mathbb{Z}_p^\ell\right\}\right)$ è una $(2^\ell - 1, p)$-famiglia hash fortemente universale.

Dimostrazione. Siano $x, x' \in \mathcal{X}$, $x \neq x'$, e sia $y, y' \in \mathbb{Z}_p$. Siccome $x \neq x'$ e $\mathbb{Z}_p$ è un campo, per il **Teorema di Rouché-Capelli**, il sistema

$$\begin{cases} r_1 x_1 + \cdots + r_\ell x_\ell = y \mod p \\ r_1 x_1' + \cdots + r_\ell x_\ell' = y' \mod p \end{cases}$$

ammette $p^{\ell-2}$ soluzioni che è esattamente $\frac{|\mathcal{K}|}{M^2}$.

□

5.12.2 Ottimalità delle Probabilità di Inganno

In questa sezione determiniamo una limitazione inferiore per le probabilità di inganno di MAC incondizionatamente sicuri e mostriamo che tale limitazione è raggiunta solo dalla famiglie hash fortemente universali.

Teorema 5.44. *Sia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ una (N, M)-famiglia hash, allora $Pd_0 \geq \frac{1}{M}$. Inoltre,*

$$Pd_0 = 1/M \iff \text{Per ogni } x \in \mathcal{X},\ y \in \mathcal{Y} \text{ vale che } |\{K \in \mathcal{K} : h_K(x) = y\}| = \frac{|\mathcal{K}|}{M}.$$

Dimostrazione. Fissato $x \in \mathcal{X}$ vale che

$$\sum_{y \in \mathcal{Y}} \text{Ricavo}(x, y) = \sum_{y \in \mathcal{Y}} \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|} = \frac{|\mathcal{K}|}{|\mathcal{K}|} = 1$$

Quindi, per ogni x esiste un y_0 dipendente da x tale che $\text{Ricavo}(x, y_0) \geq \text{Ricavo}(x, y)$ quale che sia $y \in \mathcal{Y}$. Pertanto, $\text{Ricavo}(x, y_0) \times M \geq 1$ e

$$Pd_0 \geq \text{Ricavo}(x, y_0) \geq 1/M.$$

Per ogni $x \in \mathcal{X}$, $y \in \mathcal{Y}$ vale che $|\{K \in \mathcal{K} : h_K(x) = y\}| = \frac{|\mathcal{K}|}{M}$ se e solo se

$$\text{Ricavo}(x, y) = \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|} = \frac{|\mathcal{K}|}{M} \cdot \frac{1}{|\mathcal{K}|} = \frac{1}{M}$$

che è equivalente a $Pd_0 = 1/M$.

□

Teorema 5.45. *Sia $(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$ una (N,M)-famiglia hash, allora $Pd_1 \geq \frac{1}{M}$.*

Dimostrazione. Siano $x,x' \in \mathcal{X}$ e $y \in \mathcal{Y}$ tali che $x \neq x'$ e $(x,y) \in \mathcal{V}$.

$$\begin{aligned}\sum_{y' \in \mathcal{Y}} \text{Ricavo}(x',y';x,y) &= \sum_{y' \in \mathcal{Y}} \frac{|\{K \in \mathcal{K} : h_K(x)=y, h_K(x')=y'\}|}{|\{K \in \mathcal{K} : h_K(x)=y\}|} \\ &= \frac{\sum_{y' \in \mathcal{Y}} |\{K \in \mathcal{K} : h_K(x)=y, h_K(x')=y'\}|}{|\{K \in \mathcal{K} : h_K(x)=y\}|} \\ &= \frac{|\{K \in \mathcal{K} : h_K(x)=y\}|}{|\{K \in \mathcal{K} : h_K(x)=y\}|} = 1.\end{aligned}$$

Quindi esiste $y'_0 \in \mathcal{Y}$ tale che

$$\text{Ricavo}(x',y'_0;x,y) \geq \text{Ricavo}(x',y';x,y)$$

e quindi $\text{Ricavo}(x',y'_0;x,y) \times M \geq 1$. Pertanto,

$$Pd_1 \geq \text{Ricavo}(x',y'_0;x,y) \geq 1/M.$$

□

Lemma 5.46. *Sia $(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$ una (N,M)-famiglia hash. Se $Pd_1 = 1/M$, allora per ogni $x,x' \in \mathcal{X}$ e $y \in \mathcal{Y}$ tali che $x \neq x'$ e $(x,y) \in \mathcal{V}$, risulta che $\text{Ricavo}(x',y';x,y) = 1/M$.*

Dimostrazione. Se $Pd_1 = 1/M$, segue che $\text{Ricavo}(x',y';x,y) \leq 1/M$. D'altra parte

$$\sum_{y' \in \mathcal{Y}} \text{Ricavo}(x',y';x,y) = 1$$

con $|\mathcal{Y}| = M$, pertanto $\text{Ricavo}(x',y';x,y) = 1/M$.

□

Teorema 5.47. *Sia $(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$ una (N,M)-famiglia hash. Allora*

$$Pd_1 = 1/M \iff (\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H}) \text{ è fortemente universale}$$

Dimostrazione. Se $(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$ è fortemente universale, allora $Pd_1 = 1/M$ per il **Teorema 5.41**.

Supponiamo ora che $Pd_1 = 1/M$ e proviamo che $\mathcal{V} = \mathcal{X} \times \mathcal{Y}$. Sia $(x',y') \in \mathcal{X} \times \mathcal{Y}$ e sia ora $x \in \mathcal{X}$ tale che $x \neq x'$. Sia $y \in \mathcal{Y}$ tale che $(x,y) \in \mathcal{V}$. Siccome $Pd_1 = 1/M$ allora $\text{Ricavo}(x',y';x,y) = 1/M$ per il **Lemma 5.46**.

Così

$$\frac{|\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}|}{|\{K \in \mathcal{K} : h_K(x) = y\}|} = \frac{1}{M} \tag{5.1}$$

per ogni $x, x' \in \mathcal{X}$ e $y, y' \in \mathcal{Y}$ tale che $x \neq x'$ e $(x, y) \in \mathcal{V}$. Pertanto,

$$0 < |\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}| \leq |\{K \in \mathcal{K} : h_K(x') = y'\}|$$

e quindi $(x', y') \in \mathcal{V}$. Ciò dimostra $\mathcal{V} = \mathcal{X} \times \mathcal{Y}$.
Nella dimostrazione precedente scambiando i ruoli di (x, y) e (x', y') si ha che

$$|\{K \in \mathcal{K} : h_K(x) = y\}| = |\{K \in \mathcal{K} : h_K(x') = y'\}|$$

Pertanto $|\{K \in \mathcal{K} : h_K(x) = y\}|$ è costante. Allora

$$\sum_{y \in \mathcal{Y}} \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|} = \frac{|\mathcal{K}|}{|\mathcal{K}|} = 1$$

implica $M \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|} = 1$ e quindi

$$|\{K \in \mathcal{K} : h_K(x) = y\}| = \frac{|\mathcal{K}|}{M}. \tag{5.2}$$

Sostituendo (5.1) in (5.2) segue che

$$|\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}| = \frac{|\mathcal{K}|}{M^2},$$

ovvero che $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è fortemente universale.

□

Corollario 5.48. *Sia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ una (N, M)-famiglia hash t.c. $Pd_1 = \frac{1}{M}$, allora $Pd_0 = \frac{1}{M}$.*

Dimostrazione. Sia $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ una (N, M)-famiglia hash tale che $Pd_1 = \frac{1}{M}$, allora $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ è fortemente universale per il **Teorema 5.47**, e quindi $Pd_0 = \frac{1}{M}$ per il **Teorema 5.41**.

□