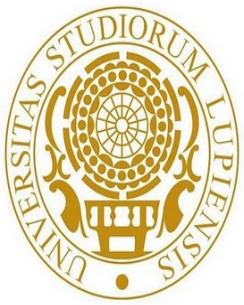
Università del Salento
Dipartimento di Matematica e Fisica
"Ennio de Giorgi"

Alessandro Montinaro
Pierluigi Rizzo

Introduzione alla Crittografia



Quaderno 1/2019
Università del Salento - Coordinamento SIBA

Quaderni di Matematica

Una pubblicazione a cura del

Dipartimento di Matematica e Fisica "Ennio de Giorgi"

Università del Salento

I Quaderni del Dipartimento di Matematica e Fisica "Ennio De Giorgi" della Università del Salento documentano gli aspetti di rilievo dell'attività di ricerca e didattica del Dipartimento. Nei Quaderni sono pubblicati articoli di carattere matematico che siano:

(1) lavori di rassegna e monografie su argomenti di ricerca;
(2) testi di seminari di interesse generale, tenuti da docenti o ricercatori del Dipartimento o esterni;
(3) lavori di specifico interesse didattico.

La pubblicazione dei lavori è soggetta all'approvazione del Comitato di Redazione, che decide tenendo conto del parere di un *referee*, nominato di volta in volta sulla base delle competenze specifiche.

Quaderno 1/2019: e-ISBN 978-88-8305-151-7

Università del Salento - Coordinamento SIBA

Indice

Introduzione

In questo quaderno, gli autori presentano gli argomenti sviluppati nelle lezioni tenute dal professore Alessandro Montinaro nel corso di Crittografia per gli studenti della Laurea Magistrale in Matematica dell'Università del Salento nell'a.a. 2018-2019.

Come il termine stesso suggerisce, la Crittogrfia è la disciplina che si occupa di "nascondere i messaggi", ovvero non di occultarli, ma di renderli incomprensibili a persone non autorizzate a leggerli. In passato, la Crittografia è stata utilizzata soprattutto in ambito militare. Oggi, invece, tutti noi facciamo uso quotidiano della Crittografia, anche senza rendercene conto. Lo sviluppo della comunicazione telematica e l'utilizzo crescente di internet per effettuare operazioni delicate (come, ad esempio, operazioni bancarie) ha fatto sì che il problema della segretezza diventasse sempre più importante.

La storia della Crittografia permette di capire come i metodi crittografici si siano evoluti nei secoli, attraverso conoscenze matematiche sempre più avanzate, intrecciandosi sempre di più con le nuove tecnologie e con il mondo dell'informatica. Durante le lezioni, infatti, si è cercato di dare importanza sia all'aspetto matematico della Crittografia, fornendo nozioni provenienti da diverse aree matematiche, come la probabilità discreta o la teoria dei numeri, sia all'aspetto computazionale, analizzando gli algoritmi utilizzati nei crittosistemi, con particolare riguardo alla loro complessità computazionale.

Nei primi capitoli di queste note, sono analizzate le tappe più importanti dello sviluppo dei sistemi crittografici dal I secolo a.C. fino alle tecniche utilizzate durante la seconda guerra mondiale. Successivamente, viene introdotta una particolare classe di cifrari a blocchi: i cifrari iterati, e viene descritto dettagliatamente il crittosistema AES (Advanced Encryption Standard), utilizzato come standard dal governo degli Stati Uniti d'America a partire dal 2001. In seguito, vengono analizzate le Funzioni Hash Crittografiche e il loro utilizzo per assicurare l'integrità dei dati. Un'ampia sezione è dedicata allo studio del crittosistema RSA, concentrandosi principalmente su tre aspetti: l'implementazione del crittosistema, l'efficienza della cifratura e il problema della sicurezza. Sono stati analizzati, infatti, i principali metodi di attacco al Crittosistema RSA, come l'Algoritmo Rho di Pollard, l'Algoritmo di Dixon per i quadrati casuali per la fattorizzazione dei moduli e, infine, l'Algoritmo di Wiener per la determinazione dell'esponente di decifratura. Successivamente, è stato introdotto il Problema del Logaritmo Discreto, alla base della sicurezza di numerosi crittosistemi. In seguito, sono state introdotte le curve ellittiche e la loro applicazione alla Crittografia nella costruzione degli analoghi dei crittosistemi basati sul Problema del Logaritmo Discreto, e l'Algoritmo di Lenstra utile per fattorizzare un intero composto dispari e quindi come ulteriore metodo di attacco al Crittosistema RSA. La parte finale del quaderno è dedicata allo studio di vari sistemi di firma

digitale, come il sistema di firma di ElGamal e le sue varianti, il sistema di firma one-time di Lamport e, infine, i sistemi di firma non ripudiabili di Chaum - Van Anterwerpen e fail-stop di Pedersen - Van Heyst.

Lecce, giugno 2019.

Alessandro Montinaro
Dipartimento di Matematica e Fisica "Ennio De Giorgi"
alessandro.montinaro@unisalento.it

Pierluigi Rizzo
Dipartimento di Matematica e Fisica "Ennio De Giorgi"
pierluigi.rizzo@unisalento.it

1 Crittosistemi Classici

1.1 Cifrari a blocchi

L'obiettivo della **Crittografia** è permettere a due utenti A e B di comunicare attraverso un canale insicuro, senza che un terzo utente non autorizzato riesca a capire l'oggetto della loro comunicazione.

Per fare ciò, A cifra il messaggio in modo tale che solo il destinatario prestabilito B possa invertire la procedura di cifratura e ricavare il messaggio originario.

Queste idee sono descritte in modo funzionale nella seguente definizione.

Definizione 1.1. (Crittosistema).
Una quintupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ di insiemi finiti e non vuoti tali che per ogni $K \in \mathcal{K}$ esistono $e_K \in \mathcal{E}$ e $d_K \in \mathcal{D}$ dove $e_K : \mathcal{P} \to \mathcal{C}$ e $d_K : \mathcal{C} \to \mathcal{P}$ e $d_K \circ e_K = Id_{\mathcal{P}}$, si dice **cifrario a blocchi**.

- $\mathcal{P}$ è l'insieme delle **unità dei messaggi in chiaro**;
- $\mathcal{C}$ è l'insieme delle **unità dei messaggi cifrati**;
- $\mathcal{K}$ è l'insieme delle **chiavi**.

Definizione 1.2. Un crittosistema si dice **endomorfico** se $\mathcal{P} = \mathcal{C}$.

La proprietà principale è $d_K \circ e_K = Id_{\mathcal{P}}$, ovvero l'**iniettività** della funzione e_K. Essa è importante al fine di evitare ambiguità nella decifratura di un messaggio. In particolare, se il sistema è endomorfico, allora e_K è una permutazione di $\mathcal{P}$.

Il mittente A del messaggio, e B, il destinatario, impiegano il seguente protocollo per usare uno specifico crittosistema:

- A e B scelgono una chiave $K \in \mathcal{K}$. Lo scambio della chiave deve avvenire attraverso un canale sicuro;
- Si supponga che A voglia mandare il messaggio in chiaro $X = x_1 x_2 ... x_n$, dove $x_i \in \mathcal{P}$, $i = 1, 2, ...n$, attraverso un canale di comunicazione insicuro; quindi A opera come segue: per $i = 1, ..., n$ calcola $y_i = e_K(x_i) \in \mathcal{C}$, con e_K algoritmo di cifratura, e trasmette $Y = y_1 ... y_n$. Un terzo individuo (l'attaccante) legge il messaggio cifrato e tenta di trovare il presunto messaggio in chiaro X' e la presunta chiave K';
- B riceve Y e per $i = 1, ...n$ calcola $x_i = d_K(y_i)$, con d_K algoritmo di decifratura, e quindi ottiene il messaggio in chiaro X.

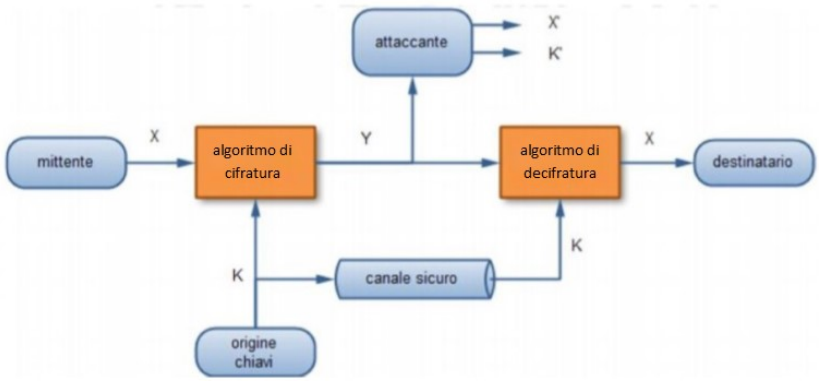


Figura 1.1: Funzionamento del protocollo di un crittosistema.

All'interno dei cifrari a blocchi distinguiamo quelli la cui chiave consta di un solo carattere, detti **monoalfabetici**, e quelli che hanno chiave composta da almeno 2 caratteri, detti **polialfabetici**.

Di seguito sono riportati alcuni cifrari classici e le loro proprietà.

1.2 Cifrario mediante sostituzione

Definizione 1.3. **(Cifrario mediante sostituzione).**
Siano $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$ e siano

$$e_K(x) = (x + K) \ mod \ 26$$
$$d_K(y) = (y - K) \ mod \ 26$$

con $x, y \in \mathbb{Z}_{26}$.

Un esempio di cifrario mediante sostituzione è il **cifrario di Cesare**. Esso prende il nome da Gaio Giulio Cesare, che lo utilizzava per proteggere i suoi messaggi segreti.
Grazie allo storico Svetonio sappiamo che Cesare utilizzava in genere una chiave di 3 per il cifrario, cioè ogni lettera viene sostituita con quella posizionata nell'alfabeto tre posti in avanti.
L'imperatore romano, in realtà, utilizzò l'alfabeto latino, ovvero $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{21}$ con $e_3(x) = (x + 3) \ mod \ 21$ e $d_3(y) = (y - 3) \ mod \ 21$.

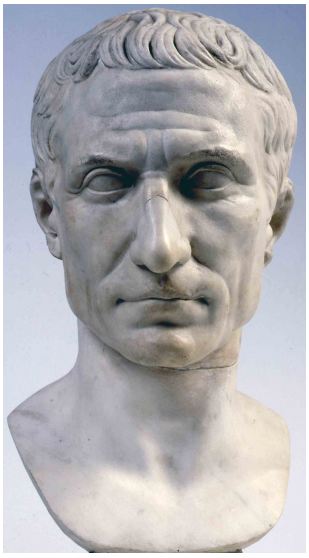


Figura 1.2: Gaio Giulio Cesare (101 a.C. – 44 a.C.)

$\mathcal{P}$	A	B	C	D	E	F	G	H	I	J	K	L	M
$\mathcal{C}$	D	E	F	G	H	I	J	K	L	M	N	O	P

$\mathcal{P}$	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$\mathcal{C}$	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Si veda come esempio la cifratura del testo **venividivici**. Si assegna ad ogni lettera un valore numerico: ad A viene associato 0 fino a Z a cui viene associato 25.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Sostituire una lettera con quella che si trova tre posti più avanti significa sommare il valore numerico associato alla lettera a 3 e poi ridurre modulo 26.
Pertanto il messaggio **venividivici** corrisponde a 21 4 13 8 21 8 3 8 21 8 2 8 che cifrato risulta 24 7 16 11 24 11 6 11 24 11 5 11, ovvero convertendo i numeri nuovamente in lettere si ha **YHQLYLQGYLFL**.

1.3 Cifrario Affine

Il cifrario mediante sostituzione è un particolare caso di cifrario affine, dove le chiavi possibili sono 26. In generale, il cifrario affine è definito come segue:

Definizione 1.4. **(Cifrario affine).**
Siano $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ e $\mathcal{K} = \{(a,b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} : mcd(a,26) = 1\}$. Se $K = (a,b) \in \mathcal{K}$, siccome $mcd(a,26) = 1$ quindi a è invertibile in $\mathbb{Z}_{26}$, allora

$$e_K(x) = (ax + b) \ mod \ 26$$
$$d_K(y) = a^{-1}(y - b) \ mod \ 26$$

con $x, y \in \mathbb{Z}_{26}$.

Si noti che

$$| \mathcal{K} | = 26 \cdot \varphi(26) = 26 \cdot 12 = 312,$$

dove φ denota la phi di Eulero.

Di seguito viene fornito un esempio di cifrario affine. Se $K = (7,3)$, sapendo che $(7^{-1}) \ mod \ 26 = 15$, allora

$$e_K(x) = (7x + 3) \quad mod \ 26$$
$$d_K(x) = (15y - 19) \quad mod \ 26$$

Si consideri come testo da cifrare **ciao**, che corrisponde a 2 8 0 14.
Cifrando si ha:

$$(7 \cdot 2 + 3) \ mod \ 26 = 17 \ mod \ 26 = 17$$
$$(7 \cdot 8 + 3) \ mod \ 26 = 59 \ mod \ 26 = 7$$
$$(7 \cdot 0 + 3) \ mod \ 26 = 3 \ mod \ 26 = 3$$
$$(7 \cdot 14 + 3) \ mod \ 26 = 101 \ mod \ 26 = 23.$$

Quindi il messaggio cifrato è **MHDX**. Applicando la funzione di decifratura si ottiene:

$$(15 \cdot 17 - 19) \ mod \ 26 = 236 \ mod \ 26 = 2$$
$$(15 \cdot 7 - 19) \ mod \ 26 = 86 \ mod \ 26 = 8$$
$$(15 \cdot 3 - 19) \ mod \ 26 = 26 \ mod \ 26 = 0$$
$$(15 \cdot 23 - 19) \ mod \ 26 = 326 \ mod \ 26 = 14,$$

che risulta essere proprio il messaggio in chiaro di partenza.

1.4 Cifrario mediante permutazione

Il cifrario mediante permutazione permette di aumentare la cardinalità dell'insieme delle chiavi, quindi la sicurezza, poiché la funzione di cifratura non è necessariamente un'applicazione affine, ma bensì una permutazione.

Definizione 1.5. (Cifrario mediante permutazione).
Sia $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ e $\mathcal{K} = Sym(26)$. Per ogni permutazione $\pi \in K$ siano

$$e_K(x) = \pi(x)$$
$$d_K(x) = \pi^{-1}(x)$$

con π^{-1} l'inversa di π.

Nel cifrario mediante permutazione è più comodo utilizzare direttamente le lettere piuttosto che convertirle nel loro valore numerico.
La chiave di un cifrario di permutazione consiste in una permutazione dei 26 caratteri. Pertanto

$$|\mathcal{K}| = |Sym(26)| = 26! = 4.0329146 \cdot 10^{26},$$

un numero molto elevato anche per un computer.

Si consideri il seguente esempio: sia

$$\pi = (0\ 15)(1\ 3\ 11\ 25\ 14\ 23\ 9\ 10\ 4\ 5\ 6\ 20\ 7\ 17\ 19\ 2\ 16\ 8\ 13\ 18\ 12\ 21\ 24\ 22)$$

la chiave del cifrario e sia **Ho bisogno di aiuto**, cioè **hobisognodiaiuto**, il testo in chiaro. Ciò numericamente è descritto come

$$7\ 14\ 1\ 8\ 18\ 14\ 6\ 13\ 14\ 3\ 8\ 0\ 8\ 20\ 19\ 14$$

che, attraverso la permutazione π, è cifrato in

$$17\ 23\ 3\ 13\ 12\ 23\ 20\ 18\ 23\ 11\ 13\ 15\ 13\ 7\ 2\ 23$$

cioè in **RXDNMXUSXLNPNHCX**.

1.5 Cifrario di Vigenère

Nel 1586 un diplomatico, traduttore e alchimista francese di nome Blaise de Vigenère rese pubblico un primo esempio di cifrario polialfabetico.

Figura 1.3: Blaise de Vigenère (1523 – 1596)

Il principale punto di forza di questo metodo è l'utilizzo non di uno ma di ben 26 alfabeti (pari al numero delle lettere dell'alfabeto inglese) per cifrare un solo messaggio. Il metodo si può quindi considerare una generalizzazione del cifrario di Cesare.
L'idea è piuttosto semplice: invece di spostare sempre dello stesso numero di posti la lettera da cifrare, come accade nel cifrario di Cesare, questa viene spostata di un numero di posti variabile, determinato in base a una parola chiave da concordarsi tra mittente e destinatario.
Per rendere più semplice il processo di cifratura, Vigenére propose l'utilizzo della **Tabella 1** composta da alfabeti ordinati e traslati.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tabella 1: Tavola di Vigenère.

Il seguente esempio e la **Tabella 1** permettono di capire il funzionamento del cifrario di Vigènere.

Si consideri un messaggio del tipo **Tu non puoi passare**, quindi si ha **tunonpuoipassare**, con la parola chiave **ISUFI**. Per cifrare la prima lettera **T** con la corrispondente lettera della chiave (in questo caso **I**) si individua l'elemento posizionato nella colonna **T** e riga **I**, che è proprio **B**.

Numericamente il messaggio precedente corrisponde a

19 20 13 14 13 15 20 14 8 15 0 18 18 0 17 4

e la parola chiave corrisponde a 8 18 20 5 8. Si ripeta la chiave più volte e si sommi modulo 26 il suo valore a quello da sostituire.

t	u	n	o	n	p	u	o	i	p	a	s	s	a	r	e	+
i	s	u	f	i	i	s	u	f	i	i	s	u	f	i	i	=

Ciò corrisponde numericamente a

19	20	13	14	13	15	20	14	8	15	0	18	18	0	17	4	+
8	18	20	5	8	8	18	20	5	8	8	18	20	5	8	8	=
1	12	7	19	21	23	12	8	13	23	8	10	12	5	25	12	

con somma in $\mathbb{Z}_{26}$. Convertendo i numeri in lettere si ha

BMHTVXMINXIKMFZM.

È evidente dall'esempio che la stessa lettera di un testo nel cifrario di Vigenère può essere cifrata con lettere diverse, inoltre lettere diverse sono cifrate con la stessa lettera.

Definizione 1.6. Sia m un intero positivo e siano $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$. In corrispondenza della chiave $K = (k_1, k_2, k_3, \cdots, k_m)$ siano

$$e_K(x_1, x_2, \cdots, x_m) = (x_1 + k_1, x_2 + k_2, \cdots, x_m + k_m)$$
$$d_K(y_1, y_2, \cdots, y_m) = (y_1 - k_1, y_2 - k_2, \cdots, y_m - k_m)$$

dove la somma è da intendersi in $\mathbb{Z}_{26}$.

Il numero di possibili chiavi di lunghezza m nel cifrario di Vigenère è 26^m, abbastanza elevato per un valore relativamente piccolo di m, e quindi la ricerca esaustiva della chiave può richiedere molto tempo.

K viene detta anche **verme** per il motivo che, se è molto più corta del messaggio, deve essere ripetuta, eventualmente troncandola, un numero di volte pari alla lunghezza del testo in chiaro.

1.6 Cifrario di Hill

Un altro esempio di cifrario polialfabetico è il cifrario di Hill, inventato dall'omonimo matematico statunitense nel 1929.

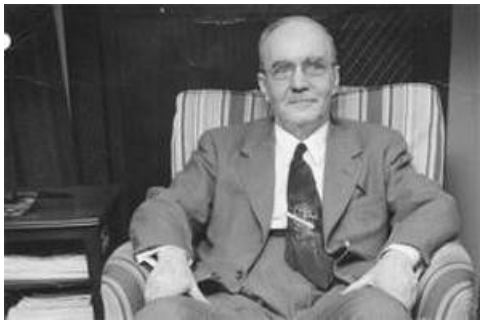


Figura 1.4: Lester S. Hill (1891 – 1961)

Definizione 1.7. **(Cifrario di Hill)**
Sia $m \geq 2$ un intero, siano $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ e sia $\mathcal{K}$ l'insieme delle matrici quadrate invertibili di ordine m a coefficienti in $\mathbb{Z}_{26}$. Se K è una chiave, allora

$$e_K(x) = xK$$
$$d_K(y) = yK^{-1}$$

dove il prodotto righe per colonne tra matrici è in $\mathbb{Z}_{26}$.

Per esempio, sia **Il tesoro luccicante** il testo in chiaro e siano $m = 6$ e la chiave definita da

$$K = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Si convertano le varie lettere nei rispettivi numeri

i	l	t	e	s	o	r	o	l	u	c	c	i	c	a	n	t	e
8	11	19	4	18	14	17	14	11	20	2	2	8	2	0	13	19	4

e si dividano i numeri così trovati in gruppi di lunghezza 6 applicando poi la funzione e_K.

$$\begin{pmatrix}8 & 11 & 19 & 4 & 18 & 14\end{pmatrix}\begin{pmatrix}0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\end{pmatrix} = \begin{pmatrix}19 & 18 & 8 & 14 & 4 & 11\end{pmatrix}$$

e cosí via per tutti i gruppi, fino ad arrivare ad avere numericamente

$$19\ 18\ 8\ 14\ 4\ 11\ 11\ 2\ 17\ 2\ 20\ 14\ 0\ 19\ 8\ 4\ 13\ 2$$

cioè **TSIOELLCRCUOATIENC**.
Utilizzando la matrice inversa

$$K^{-1} = \begin{pmatrix}0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\end{pmatrix}$$

è facile vedere che si ottiene il messaggio di partenza.

1.7 Cifrario mediante trasposizione

L'idea su cui si basa il cifrario mediante trasposizione è quella di non modificare le lettere, ma solo la loro posizione attraverso l'utilizzo di permutazioni.

Definizione 1.8. **(Cifrario mediante trasposizione)**
Siano $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ e $\mathcal{K} = Sym(m)$, dove $m \in \mathbb{N}$. Se $\pi \in \mathcal{K}$, allora

$$e_\pi(x_1, ..., x_m) = (x_{\pi(1)}, ..., x_{\pi(m)})$$
$$d_\pi(y_1, ..., y_m) = (y_{\pi^{-1}(1)}, ..., y_{\pi^{-1}(m)}),$$

dove π^{-1} è la permutazione inversa di π.

Osserviamo che il cifrario di trasposizione è un caso particolare del cifrario di Hill. Infatti, a una permutazione π dell'insieme $\{1, \cdots, m\}$ possiamo associare **una matrice di permutazione** K_π in cui il termine di posto $a_{ij} = 1$ se, e solo se, $\pi(i) = j$, e 0 altrimenti.

Riprendendo l'esempio precedente, la chiave

$$K_\pi = \begin{pmatrix} 0&0&1&0&0&0 \\ 0&0&0&0&0&1 \\ 1&0&0&0&0&0 \\ 0&0&0&0&1&0 \\ 0&1&0&0&0&0 \\ 0&0&0&1&0&0 \end{pmatrix}$$

è la matrice della permutazione $\pi = (1\ 3)(2\ 6\ 4\ 5)$. Inoltre, $K_\pi^{-1} = K_{\pi^{-1}}$.

1.8 Cifrari a flusso

Nei cifrari a blocchi la chiave è costante, cioè $y = y_1 \cdots y_n = e_K(x_1), \cdots e_K(x_n)$. Un'idea differente è quella alla base dei **cifrari a flusso** in cui è possibile generare un flusso di chiavi $z = z_1 z_2 ... z_n$ e utilizzare queste per cifrare il messaggio attraverso la regola

$$y = y_1 y_2 \cdots = e_{z_1}(x_1) e_{z_2}(x_2) \cdots$$

Esistono diversi tipi di cifrari a flusso. Il più semplice è quello in cui il flusso di chiavi viene costruito a partire da una chiave indipendente dal testo da decifrare.

Definizione 1.9. **(Cifrario a flusso sincrono)**
Un **cifrario a flusso sincrono** è una sestupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{E}, \mathcal{D})$ di insiemi finiti non vuoti e una funzione $g : \mathcal{K} \to \mathcal{L}^{\mathbb{N}}$ tali che

- $\mathcal{P}$ è l'insieme delle **unità di messaggio in chiaro**;
- $\mathcal{C}$ è l'insieme delle **unità di messaggio cifrato**;
- $\mathcal{K}$ è l'insieme delle **chiavi**;
- $\mathcal{L}$ è l'insieme detto **alfabeto delle chiavi**;
- g è il **generatore di chiavi**: una funzione che ad ogni chiave $k \in K$ associa una stringa infinita $z_1 z_2 \cdots$ detta **flusso delle chiavi**, $z_i \in \mathcal{L}$ per ogni $i \geq 1$;
- per ogni $z \in \mathcal{L}$ c'è una **funzione di cifratura** $e_z \in \mathcal{E}$, $e_z : \mathcal{P} \to \mathcal{C}$ e una **funzione di decifratura** $d_z \in \mathcal{D}$, $d_z : \mathcal{C} \to \mathcal{P}$ t.c. $d_z \circ e_z = Id_{\mathcal{P}}$.

Esempi di cifrari a flusso sincrono sono:

- I **cifrari a blocchi**, per cui vale che $\mathcal{L} = \mathcal{K}$ e $g : \mathcal{K} \to \mathcal{K}^{\mathbb{N}}$, $g(K) = (K, K, \cdots)$ è la successione costante di costante valore K;
- Il **cifrario di Vigenère** con $\mathcal{K} = (\mathbb{Z}_{26})^m$, $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathbb{Z}_{26}$ e il flusso $g : (\mathbb{Z}_{26})^m \to (\mathbb{Z}_{26})^{\mathbb{N}}$ di chiavi definito come segue:

$$z_i = \begin{cases} k_i \text{ se } 1 \leq i \leq m \\ z_{i-m} \text{ se } i \geq m+1 \end{cases}$$

dove $K = (k_1 \cdots k_m)$. Quindi, si ha

$$e_z(x) = (x + z) \mod 26$$

$$d_z(y) = (y - z) \mod 26.$$

Un cifrario a flusso è detto **periodico** con periodo d se $z_{i+d} = z_i$ per tutti gli interi $i \geq 1$. Un cifrario a flusso è spesso descritto in termini di alfabeto binario, cioè $\mathcal{P} = \mathcal{C} = \mathbb{Z}_2$. Cifratura e decifratura sono operazioni in $\mathbb{Z}_2$:

$$e_z(x) = (x + z) \mod 2$$
$$d_z(y) = (y + z) \mod 2$$

Questo tipo di operazioni sono molto efficienti nell'hardware ed è per questo che questo cifrario viene implementato spesso.

Sia $K = (k_1, ..., k_m, c_0, ..., c_{m-1}) \in \mathbb{Z}_2^{2m}$ con $c_0, c_1, ..., c_{m-1}$ opportune costanti e sia $z_i = k_i$ con $1 \leq i \leq m$. Usando una ricorrenza lineare di grado m il flusso di chiavi è

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \mod 2$$

per tutti gli $i \geq 1$.

A partire dalle $2m$-uple $(k_1 \cdots k_m, c_0 \cdots c_{m-1})$ in cui $c_0, ..., c_{m-1}$ sono scelte in maniera opportuna viene creato un flusso di chiavi di periodo $2^m - 1$.
Sia per esempio $m = 4$ e il flusso di chiavi sia generato usando la ricorrenza lineare

$$z_{i+4} = (z_i + z_{i+1}) \mod 2$$

con $i \geq 1$. Se il flusso è inizializzato con un vettore diverso da $(0, 0, 0, 0)$, allora otteniamo un flusso di periodo $2^4 - 1 = 15$. Per esempio, iniziando con $(1, 0, 0, 0)$ si avrà

$$1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1 \cdots$$

Un qualsiasi altro vettore darà un permutazione ciclica dello stesso flusso di chiavi.

Un altro metodo di generazione di flusso di chiavi è **linear feedback shift register (LFSR)**.

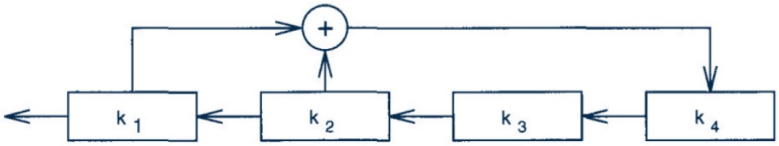


Figura 1.5: Esempio di LFSR

Come evidenziato in **Figura 1.5**, il vettore binario $(k_1, \cdots, k_m)$ è utilizzato per inizializzare il flusso e ad ogni passo vengono applicate le seguenti operazioni:

- k_1 viene messo in cima come successivo bit del flusso di chiavi;
- ognuno dei $k_2 \cdots k_m$ viene spostato a sinistra di un posto;
- il "nuovo" valore di k_m inserito è

$$\sum_{j=0}^{m-1} c_j k_{j+1}.$$

LSFR è eseguito per un certo numero di passi e utilizza una somma in $\mathbb{Z}_2$.

Un **cifrario a flusso non sincrono** è un cifrario in cui il flusso di chiavi z_i dipende dalle precedenti unità di messaggi in chiaro così come dalla chiave K.

Definizione 1.10. **(Cifrario autokey)**
Siano $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$. Sia $g : \mathbb{Z}_{26} \to (\mathbb{Z}_{26})^{\mathbb{N}}$ definito da $z_1 = K$ e $z_i = x_{i-1}$ per tutti gli $i \geq 2$. Per $0 \leq z \leq 25$ siano

$$e_z(x) = (x + z) \ mod \ 26$$
$$d_z(y) = (y - z) \ mod \ 26$$

con $x, y \in \mathbb{Z}_{26}$.

Si osservi che, dal punto di vista formale, il cifrario autokey sembra molto vicino a quello di Vigenère. La ragione per cui viene denominato in questo modo è dovuto all'autogenesi della chiave durante la cifratura del messaggio. Il cifrario autokey è pertanto insicuro perché vi sono solo 26 possibili chiavi.

Sia $K = 8$ la chiave e il testo da cifrare **buona giornata**. Convertito in interi:

1 20 14 13 0 6 8 14 17 13 0 20 0

il flusso di chiavi sarà il seguente:

8 1 20 14 13 0 6 8 14 17 13 0 20

sommandole modulo 26 si ha:

9 21 8 1 13 6 14 22 5 4 13 20 20

che convertito in lettere è **JVIBNGOWFENUU**.

1.9 La macchina Enigma

Enigma era una macchina per cifrare utilizzata dal Terzo Reich negli anni precedenti e durante la Seconda Guerra Mondiale.

La macchina era costituita da:

- una **tastiera** per immettere il testo da cifrare;
- lo **scambiatore** che cifra la lettera del testo in chiaro nella corrispondente lettera del testo cifrato;
- un **visore** con varie lampadine che illuminandosi indicano la lettera del testo cifrato da trasmettere.

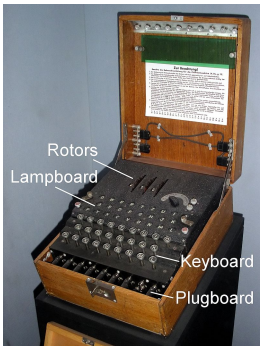


Figura 1.6: Macchina Enigma

Lo scambiatore (o rotore) rappresenta la parte più importante della macchina. Consiste in uno spesso disco di gomma attraversato da una fitta rete di fili proveniente dalla tastiera, che entrano nello scambiatore ed escono dalla parte opposta, determinando una cifratura a sostituzione monoalfabetica.

L'idea fondamentale di Scherbius, uno dei due creatori di Enigma, fu quella di far ruotare il disco di un ventiseiesimo di giro, in modo tale da cambiare cifratura ad ogni lettera pigiata sulla tastiera trasformando così la cifratura in una sostituzione polialfabetica.

Figura 1.7: Arthur Scherbius (1878 – 1929)

Per aumentare la sicurezza di Enigma vennero poi inseriti un secondo e un terzo scambiatore: il secondo compiva una rotazione parziale soltanto dopo che il primo avesse compiuto un intero giro e il terzo compiva la stessa cosa basandosi sul secondo. Il numero totale di chiavi è circa 10 milioni di miliardi, un numero enorme anche per un computer!

Di seguito è presentata una versione semplificata del cifrario che sta alla base di Enigma.

Siano $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$ e π una fissata permutazione di $\mathbb{Z}_{26}$.
Sia $g : \mathcal{K} \to \mathcal{L}^{\mathbb{N}}$ definita da $z_i = (K + i - 1) \ mod\ 26$ per $i \geq 1$.

La cifratura e decifratura utilizzano π e π^{-1} come segue

$$e_z(x) = \pi(x) + z \ mod\ 26$$
$$d_z(x) = \pi^{-1}(y) - z \ mod\ 26,$$

dove $z \in \mathbb{Z}_{26}$.

Si consideri la seguente permutazione di $\mathbb{Z}_{26}$:

$$\pi = (0\ 23\ 9\ 16\ 17\ 2\ 24\ 3)(1\ 13\ 18\ 21\ 4\ 7\ 6\ 14\ 5\ 15\ 11)(8\ 25)(10\ 22)(12\ 19)$$

Si consideri il messaggio in chiaro **Spruchnummer** ("messaggio numero" in tedesco, una delle frasi che permise al gruppo di Bletchley Park [1] guidato da Alan Turing di decifrare il funzionamento di Enigma durante la II guerra mondiale) e si cifri seguendo il crittosistema appena definito.

[1] Bletchley Park fu il sito dell'unità principale di crittoanalisi del Regno Unito, nonchè sede della Scuola governativa di codici e cifratura durante la Seconda Guerra Mondiale. Le informazioni ottenute con le attività svolte a Bletchley Park hanno aiutato lo sforzo alleato e accorciato la durata della guerra.

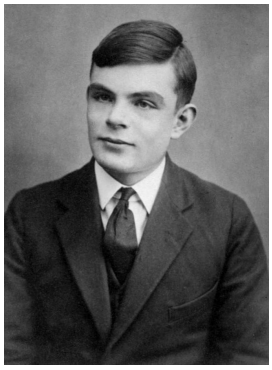
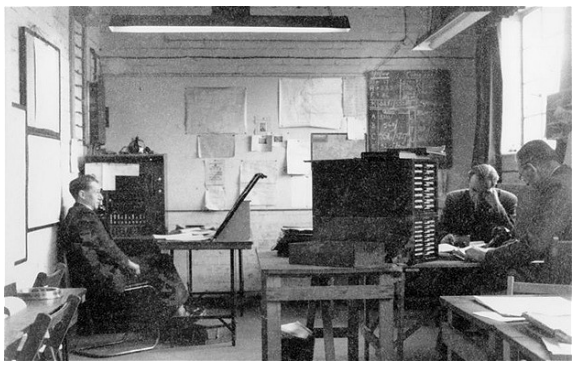



Figura 1.8: Alan Mathison Turing (1912 – 1954) e la stanza di controllo nella Baracca 6 (crittoanalisi dei codici Enigma per l'Army and Air Force) nel Bletchley Park

Quindi,

s	**p**	**r**	**u**	**c**	**h**	**n**	**u**	**m**	**m**	**e**	**r**
18	**15**	**17**	**20**	**2**	**7**	**13**	**20**	**12**	**12**	**4**	**17**

Sapendo che $K = 8$, il flusso di chiavi è:

$$
\begin{aligned}
z_1 &= (K + 1 - 1) \ mod\ 26 = 8 \\
z_2 &= (K + 2 - 1) \ mod\ 26 = 9 \\
z_3 &= (K + 3 - 1) \ mod\ 26 = 10 \\
z_4 &= (K + 4 - 1) \ mod\ 26 = 11 \\
z_5 &= (K + 5 - 1) \ mod\ 26 = 12 \\
z_6 &= (K + 6 - 1) \ mod\ 26 = 13 \\
z_7 &= (K + 7 - 1) \ mod\ 26 = 14 \\
z_8 &= (K + 8 - 1) \ mod\ 26 = 15 \\
z_9 &= (K + 9 - 1) \ mod\ 26 = 16 \\
z_{10} &= (K + 10 - 1) \ mod\ 26 = 17 \\
z_{11} &= (K + 11 - 1) \ mod\ 26 = 18 \\
z_{12} &= (K + 12 - 1) \ mod\ 26 = 19
\end{aligned}
$$

Ora, cifrando il messaggio si ha:

$$
\begin{aligned}
e_{z_1}(s) &= (\pi(S) + z_1) \ mod\ 26 = 3 \\
e_{z_2}(p) &= (\pi(P) + z_2) \ mod\ 26 = 20 \\
e_{z_3}(r) &= (\pi(R) + z_3) \ mod\ 26 = 12 \\
e_{z_4}(u) &= (\pi(U) + z_4) \ mod\ 26 = 5 \\
e_{z_5}(c) &= (\pi(C) + z_5) \ mod\ 26 = 10 \\
e_{z_6}(h) &= (\pi(H) + z_6) \ mod\ 26 = 19 \\
e_{z_7}(n) &= (\pi(N) + z_7) \ mod\ 26 = 6 \\
e_{z_8}(u) &= (\pi(U) + z_8) \ mod\ 26 = 9 \\
e_{z_9}(m) &= (\pi(M) + z_9) \ mod\ 26 = 9 \\
e_{z_{10}}(m) &= (\pi(M) + z_{10}) \ mod\ 26 = 10 \\
e_{z_{11}}(e) &= (\pi(E) + z_1 1) \ mod\ 26 = 25 \\
e_{z_{12}}(r) &= (\pi(R) + z_1 2) \ mod\ 26 = 21
\end{aligned}
$$

cioè **DUMFKTGJJKZV**.

Volendo invece decriptare lo stesso testo appena cifrato si ha:

$$
\begin{aligned}
d_{z_1}(D) &= (\pi^{-1}(D) - z_1) \ mod\ 26 = 18 \\
d_{z_2}(U) &= (\pi^{-1}(U) - z_2) \ mod\ 26 = 15 \\
d_{z_3}(M) &= (\pi^{-1}(M) - z_3) \ mod\ 26 = 17 \\
d_{z_4}(F) &= (\pi^{-1}(F) - z_4) \ mod\ 26 = 20 \\
d_{z_5}(K) &= (\pi^{-1}(K) - z_5) \ mod\ 26 = 2 \\
d_{z_6}(T) &= (\pi^{-1}(T) - z_6) \ mod\ 26 = 7 \\
d_{z_7}(G) &= (\pi^{-1}(G) - z_7) \ mod\ 26 = 13 \\
d_{z_8}(J) &= (\pi^{-1}(J) - z_8) \ mod\ 26 = 20 \\
d_{z_9}(J) &= (\pi^{-1}(J) - z_9) \ mod\ 26 = 12 \\
d_{z_{10}}(K) &= (\pi^{-1}(K) - z_{10}) \ mod\ 26 = 12 \\
d_{z_{11}}(Z) &= (\pi^{-1}(Z) - z_{11}) \ mod\ 26 = 4 \\
d_{z_{12}}(V) &= (\pi^{-1}(V) - z_{12}) \ mod\ 26 = 17
\end{aligned}
$$

che corrisponde al testo in chiaro di partenza.

2 Crittoanalisi

2.1 Principio di Kerckhoffs

Uno dei principi base della Crittografia è la seguente

Definizione 2.1. **(Principio di Kerckhoffs)**
La sicurezza di un crittosistema dipende solo dalla segretezza della chiave.

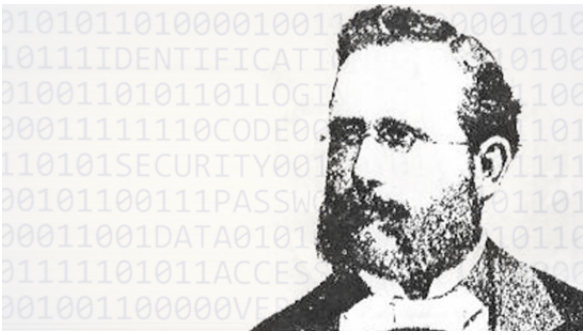


Figura 2.1: Auguste Kerckhoffs (1835 – 1903)

Esistono diversi **modelli di attacco** ai crittosistemi, però tutti hanno a comune un solo obiettivo: determinare la chiave. La differenza principale tra tali modelli è rappresentato dalle informazioni che l'avversario ha a disposizione quando esso effettua un attacco.

I modelli di attacco sono classificati come segue:

- **Attacco basato sulla conoscenza del testo cifrato**: l'avversario conosce una stringa y di testo cifrato.
- **Attacco basato sulla conoscenza del testo in chiaro**: l'avversario conosce una stringa x di testo in chiaro e la corrispondente stringa y di testo cifrato.
- **Attacco basato sulla conoscenza del testo in chiaro scelto**: l'avversario ha accesso temporaneo alla macchina di cifratura. Quindi, può scegliere un'opportuna stringa di testo in chiaro x e calcolarne la corrispondente stringa di testo cifrato y.

Esempio 2.2. Durante la II guerra mondiale gli inglesi posizionarono delle mine in certe località, sapendo che i tedeschi, trovate quelle mine, avrebbero criptato i luoghi e li avrebbero rimandati al quartier generale. Questi messaggi crittografati sono stati utilizzati dai crittoanalisti del Bletchley Park (il sito dell'unità principale di crittoanalisi del Regno Unito in quegli anni) per decifrare lo schema di crittografia tedesco.

Esempio 2.3. Nel Maggio del 1942, i crittoanalisti della Marina statunitense intercettarono un messaggio crittografato dai giapponesi che furono in grado di decodificare parzialmente.
Il risultato indicava che i giapponesi stavano pianificando un attacco su AF, dove AF era un frammento di testo cifrato che gli Stati Uniti non erano in grado di decodificare.
Per altri motivi, gli Stati Uniti credevano che l'obiettivo fossero le Isole di Midway. Sfortunatamente, i loro tentativi di convincere i pianificatori di Washington furono futili; la convinzione generale era che le Midway non potessero essere il bersaglio.
I crittoanalisti della Marina hanno escogitato il seguente piano: hanno incaricato le forze statunitensi di inviare un falso messaggio alle Midway e i giapponesi hanno immediatamente riferito ai loro superiori che *"AF è a corto di acqua"*.
I crittoanalisti della Marina avevano ora la loro prova che AF corrispondeva a Midway, e gli Stati Uniti spedirono tre portaerei in quella posizione. Il risultato fu che le Isole di Midway furono salvate ed i giapponesi subirono perdite significative. Questa battaglia fu un punto di svolta nella guerra tra Stati Uniti e Giappone nel Pacifico.
I crittoanalisti della Marina hanno effettuato un attacco con testo in chiaro, poichè erano in grado di influenzare i giapponesi (anche se in modo indiretto) per crittografare la parola "Midway". Se lo schema di crittografia giapponese fosse stato protetto contro gli attacchi con testo in chiaro scelto, questa strategia da parte dei crittoanalisti statunitensi non avrebbe funzionato (e la storia sarebbe potuta andare in modo molto diverso!).

- **Attacco basato sulla conoscenza del testo cifrato scelto**: l'avversario ha accesso temporaneo alla macchina di decifratura. Quindi, può scegliere un'opportuna stringa di testo cifrato y e calcolarne la corrispondente stringa di testo in chiaro x.

Esempio 2.4. Come nel caso degli attacchi basati sulla conoscenza del testo in chiaro scelto, non ci aspettiamo che le parti oneste decodifichino il testo cifrato arbitrario scelto da un avversario.
Tuttavia, potrebbero esserci degli scenari in cui un avversario potrebbe essere in grado di influenzare ciò che viene decodificato e apprendere alcune informazioni parziali sul risultato.

Esempio 2.5. Ad esempio:

1. Nell'**Esempio 2.3** è concepibile che anche i crittoanalisti statunitensi abbiano provato a inviare messaggi crittografati ai giapponesi e quindi a monitorare il loro comportamento. Tale comportamento (ad esempio il movimento di forze e poi simili) avrebbe potuto fornire informazioni importanti sul testo in chiaro sottostante.

2. Immagina un utente che invia messaggi crittografati alla propria banca. Un avversario può essere in grado di inviare i testi cifrati per conto di tale utente; la banca decodifica i testi cifrati e l'avversario potrebbe imparare qualcosa sul risultato. Ad esempio, se un testo cifrato corrisponde a un testo in chiaro non formattato (ad esempio un messaggio incomprensibile o semplicemente uno non formattato correttamente), l'avversario può essere in grado di dedurlo dalla reazione della banca (cioè lo schema della comunicazione successiva).

2.2 Crittoanalisi del cifrario mediante sostituzione

Il cifrario mediante sostituzione non è sicuro, poiché può essere decifrato con il **metodo della ricerca esaustiva della chiave** (o anche **forza bruta**), cioè andando a provare tutte le chiavi possibili e vedendo quale fornisce una frase di senso compiuto; infatti abbiamo solo 26 chiavi possibili (di cui una è quella banale), ed è facile determinare quella giusta provando tutte le chiavi.

Il seguente metodo può essere utilizzato per tutti i cifrari che hanno numeri non eccessivamente grandi di chiavi possibili.

Sia **XTQIFYN** il testo cifrato, convertito in numeri 23 19 16 8 5 24 13; si scelga casualmente una chiave, per esempio $K = 4$ e si applichi la funzione di decifratura corrispondente:

$$\begin{aligned} d_K(23) &= (23-4) \mod 26 = 19 \\ d_K(19) &= (19-4) \mod 26 = 15 \\ d_K(16) &= (16-4) \mod 26 = 12 \\ d_K(8) &= (8-4) \mod 26 = 4 \\ d_K(5) &= (5-4) \mod 26 = 1 \\ d_K(24) &= (24-4) \mod 26 = 20 \\ d_K(13) &= (13-4) \mod 26 = 9 \end{aligned}$$

che sarà quindi **tpmebuj** che non ha senso compiuto.

Applicando nuovamente lo stesso procedimento con $K = 5$, si avrà

$$\begin{aligned} d_K(23) &= (23-5) \mod 26 = 18 \\ d_K(19) &= (19-5) \mod 26 = 14 \\ d_K(16) &= (16-5) \mod 26 = 11 \\ d_K(8) &= (8-5) \mod 26 = 3 \\ d_K(5) &= (5-5) \mod 26 = 0 \\ d_K(24) &= (24-5) \mod 26 = 19 \\ d_K(13) &= (13-5) \mod 26 = 8 \end{aligned}$$

che convertito è **soldati** che ha senso compiuto e rappresenta quindi il messaggio in chiaro di partenza.

2.3 Crittoanalisi del cifrario affine e cifrario mediante permutazione

Per la crittoanalisi del cifrario affine si utilizza la **Tabella delle occorrenze** delle 26 lettere dell'alfabeto. È noto che in qualsiasi alfabeto esistono lettere che si ripetono più spesso di altre. Si viene a creare quindi una tabella delle probabilità con cui le lettere della lingua italiana appaiono nelle parole di senso compiuto.

a	**0,117**		**n**	**0,069**
b	**0,009**		**o**	**0,098**
c	**0,045**		**p**	**0,031**
d	**0,037**		**q**	**0,005**
e	**0,118**		**r**	**0,064**
f	**0,009**		**s**	**0,049**
g	**0,016**		**t**	**0,056**
h	**0,015**		**u**	**0,030**
i	**0,113**		**v**	**0,021**
j	**0,000**		**w**	**0,000**
k	**0,000**		**x**	**0,000**
l	**0,065**		**y**	**0,000**
m	**0,025**		**z**	**0,000**

Tabella 2: Tabella delle occorrenze.

Si cercano le lettere che appaiono più frequentemente nel testo cifrato e, osservando la tabella delle frequenze, si associano a quelle più frequenti nella lingua corrispondente, nel caso specifico italiana.

Si crea quindi un sistema $y = e_k(x) = ax + b$ dato dalle coppie (x_i, y_i), dove y_i sono le lettere del testo cifrato più frequenti e le x_i sono le presunte lettere del testo in chiaro. Lo scopo è quello di trovare $K = (a, b)$, verificando che $mcd(a, 26) = 1$. La coppia utilizzata è corretta se il testo ha senso compiuto.
Per esempio, il testo da decifrare è

XKLJSUJNXDYBUJTYOBLRGXUJDXKJWLTYXSX
ALYQXLAAJYRGXWXUXNNXKBRGJLKLVXYQXRG
XRBNXKLUJNXDYLJSJYQXKYBUXSOBL

Si cerchino le due coppie (x_1, y_1) e (x_2, y_2) tali che $ax_1 + b = y_1$ e $ax_2 + b = y_2$ per determinare $K = (a, b)$. I caratteri sono distribuiti come nella tabella seguente

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
3	**6**	**0**	**3**	**0**	**0**	**4**	**0**	**0**	**10**	**5**	**10**	**0**	**5**	**2**
P	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**				
0	**3**	**5**	**4**	**2**	**6**	**1**	**2**	**18**	**9**	**0**				

Tabella 3: Tabella delle occorrenze dell'esempio.

Rifacendoci alla tabella delle frequenze si supponga che **X** sia **e** e che **J** sia **a**. Si ha quindi $4a + b = 23$ e $a0 + b = 9$; la congruenza $4a \equiv 14 \pmod{26}$ ha due soluzioni $a = 10$ e $a = -3 = 23$. Poiché la prima è sicuramente non corretta (il massimo comun divisore è diverso da 1) proviamo con la seconda sfruttando la sostituzione $y = 23x + 9$. Sapendo che $23^{-1} \equiv 17 \pmod{26}$, si ha che $x = 17(y - 9)$ e decifrando si ottiene

eriaxfaqecvufaovhuigbeface raniovexeivpeiddavgbenefeqqerugba
iriwevpegbeguqerifaqecviaxavpervufexhui

che però sembra non corretta.

Si consideri ora la sostituzione **L** con **a**, allora $4a + b = 23$ e $a0 + b = 11$; la congruenza $4a \equiv 12 \pmod{26}$ ha due soluzioni: $a = 3$ e $a = 16$; la seconda non è sicuramente corretta mentre, provando la sostituzione $y = 3x + 11$, si ha che $x = 9(y - 11)$.

Decifrando si ha come messaggio in chiaro:

eraildisegnodiunboachedigerivaunelefanteaffinchevedess
erochiaramentechecoseradisegnailinternodelboa

cioè *"era il disegno di un boa che digeriva un elefante. Affinché vedesse chiaramente che cos'era disegnai l'interno del boa"* da *Il piccolo principe* di Antoine de Saint-Exupèry.

Il metodo di crittoanalisi per il cifrario mediante permutazione è analogo a quello del cifrario affine; la permutazione non modifica la frequenza delle singole lettere e l'analisi delle frequenze è necessariamente simile a quella di un testo in chiaro. Suddividendo, quindi, il testo cifrato in porzioni di testo si cercano parole di senso compiuto; una volta trovata la regolarità, questa può essere estesa al resto del messaggio. Pertanto un attacco di forza bruta nel caso peggiore dovrebbe provare tutte le $n!$ possibili permutazioni con n lunghezza del messaggio.

2.4 Crittoanalisi del cifrario di Vigènere

La crittoanalisi del cifrario di Vigenére si basa principalmente su due tecniche: il **Test di Kasiski** e l'**Indice di coincidenza**. Entrambe vengono utilizzate per calcolare la lunghezza della stringa che rappresenta la chiave.

Il metodo Kasiski prende il nome dal maggiore prussiano Friedrich Kasiski, che nel 1863 pubblicò un metodo di decifratura della tavola di Vigenére.

Figura 2.2: Friedrich Wilhelm Kasiski (1805 – 1881)

Il Test di Kasiski venne descritto da Friedrich Kasiski nel 1863, tuttavia venne utilizzato anche prima da Charles Babbage, intorno al 1854.

È basato sull'osservazione che **due segmenti identici di testo in chiaro vengono decriptati nello stesso modo se, e solo se, la distanza tra questi è un multiplo della lunghezza della chiave**.
Viceversa, se osserviamo che **due segmenti di testo cifrato, ognuno di lunghezza almeno 3, sono identici, è probabile che corrispondano a segmenti identici di testo in chiaro**.

Il Test di Kasiski cerca porzioni identiche di testo cifrato di lunghezza almeno 3 e memorizza la distanza tra le posizioni di partenza. Si ottengono quindi diverse distanze e vale che la lunghezza della chiave m divide il massimo comun divisore delle distanze misurate.

L'Indice di coincidenza fu introdotto nel 1920 da Wolfe Friedman.

Figura 2.3: Wolfe Frederick Friedman (1891 – 1969)

Definizione 2.6. **(Indice di coincidenza)**
Sia $x = x_1x_2\cdots x_n$ una stringa di n lettere. L'**Indice di coincidenza** di x, denotato con $I_c(x)$, è la probabilità che $x_h = x_k$ per $h \neq k$.

Siano $f_0, f_1 \cdots f_{25}$ le frequenze di A, B, ..., Z rispettivamente nel testo cifrato. Il numero di coppie di lettere uguali è $\sum_{i=0}^{25} \binom{f_i}{2}$.
Quindi l'Indice di coincidenza è

$$I_c(x) = \frac{\sum_{i=0}^{25} \binom{f_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i-1)}{n(n-1)}.$$

Vale che

$$\frac{f_i - 1}{n-1} \approx \frac{f_i}{n} \approx p_i$$

dove i p_i sono dati nella **Tabella 2**, quindi

$$I_c(x) \approx \sum_{i=0}^{25} {p_i}^2 = 0,075.$$

Pertanto l'Indice di coincidenza nel testo cifrato a partire da un testo in chiaro scritto in italiano deve essere approssimativamente $0,075$.

Ora, si supponga di avere un testo cifrato $y = y_1 y_2 ... y_n$, con $y_i \in \mathbb{Z}_{26}$ con $i \in \{1, ..., n\}$ costruito usando il cifrario di Vigenère.
Siano $y'_1\ y'_2 \cdots y'_m$ m sottostringhe di y rappresentate in colonne in una tabella di dimensioni $m \times \frac{n}{m}$:

$$\begin{array}{ccccccc} y'_1 & = & y_1 & y_{m+1} & y_{2m+1} & \cdots \\ y'_2 & = & y_2 & y_{m+2} & y_{2m+2} & \cdots \\ \vdots & & \vdots & \vdots & \vdots & \vdots \\ y'_m & = & y_m & y_{2m} & y_{3m} & \cdots \end{array}$$

Se $y'_1\ y'_2 \cdots y'_m$ sono costruiti in questo modo e m è la lunghezza della chiave, allora ogni $I_c(y'_i)$ è approssimativamente uguale a $0,075$. D'altra parte, se m non è la lunghezza della chiave, allora la sottostringa y'_i sembrerà molto più casuale e il valore di $I_c(y'_i)$ si scosterà visibilmente da $0,075$.
Si osservi che una stringa completamente casuale avrà un indice di coincidenza $I_c(x)$ approssimativamente uguale a $0,038$, per l'equiprobabilità di ogni lettera, e i due valori sono sufficientemente lontani da darci la possibilità di determinare la lunghezza della chiave grazie a questo metodo (o confermare il Test di Kasiski).

Si supponga ora di aver determinato il valore corretto di m e si voglia determinare ora la chiave. Descriviamo un metodo semplice ma efficace per fare ciò.

Sia $1 \leq i \leq m$ e siano $f_0, f_1 \cdots f_{25}$ le frequenze di A, B,...,Z nella stringa y'_i. Sia inoltre $n' = \frac{n}{m}$ la lunghezza della stringa y'_i; la distribuzione di probabilità delle 26 lettere in y'_i è

$$\frac{f_0}{n'}, \cdots, \frac{f_{25}}{n'}$$

Si noti che la sottostringa y'_i ottenuta dalla cifratura di un sottoinsieme del testo in chiaro mediante sostituzione con chiave k_i. Quindi, dobbiamo sperare che

$$\frac{f_{k_i}}{n'}, \cdots, \frac{f_{25+k_i}}{n'}$$

sia "vicina" alla distribuzione ideale con $p_0, \cdots, p_{25}$.

Supposto $0 \leq g \leq 25$ sia

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'}.$$

Se $g = k_i$ allora $M_g \approx 0,075$ simile all'indice di coincidenza. Se $g \neq k_i$, allora M_g sarà di solito significativamente più piccolo di 0,075. Questa tecnica quindi dovrebbe permetterci di determinare il corretto valore di k_i per ogni i, con $1 \leq i \leq m$.

Di seguito è fornito un testo cifrato secondo Vigènere che viene decifrato utilizzando il Test di Kasiski e l'Indice di coincidenza.

DLSVHR**LTFBLP**JVTNPTKQSAXZTWXOCTWZZKW**UPTKW**
IFGIIFEGJMLEKLVSNWLIRKUEMVTRLDALRVIUNUDXS
RTRBGOGJKJZYTQVTLFTYZXVMVLODXWEAFAADUWNAOO
MMFEUJCTTYJINLRRAGWOKIJTGVAWWBROYTGDWEAXR
KWXOJWDLYZBMLZRAMWRVKGDZVWUNOUMFEGCQVTHFZ
FPUVQDNAZVGXKSIKEGMIAYWLMAEKDXALYGIJRKIMA
WZVZJZXVI**UPTKW**DPMYMSWRZVLZXEWDLHZBSKOFV**WO**
KCTSEOXZ**WOKCTS**XGCMKTGGWKEGTWEPGHCAWGJCVTA
EIYCGEZMAKKIYWORBSLVZKUZYLTELXVIUTTHCWNKEB
GAGJAAOGCTWFRKQEPIRXSYTVLWWBZTDLMXQ
GOOXQWSGMMEBAVTD**LTFBLP**IFVLCUZTKZRZBGPXR

Per determinare la lunghezza della chiave si utilizza il Test di Kasiski. Cercando le ripetizioni nel testo si ha che

- il gruppo di lettere **LTFBLP** è ripetuto nel testo a distanza di 415 lettere;
- il gruppo di lettere **UPTKW** è ripetuto nel testo a distanza di 220 lettere;
- il gruppo di lettere **WOKCTS** è ripetuto a distanza di 10 lettere.

Quindi è ipotizzabile che la lunghezza delle chiave sia

$$mcd(415, 220, 10) = 5.$$

Sapendo la lunghezza della chiave, le lettere nelle posizioni $n + 5k$ dove $n \in \{0, 1, 2, 3, 4\}$ sono cifrate con lo stesso cifrario mediante sostituzione. Analizzando le frequenze dei gruppi di lettere e confrontandoli con l'analisi delle frequenze nelle lettere italiane, è possibile capire quale è la sostituzione delle lettere. Per esempio, nel caso delle lettere di posto $1 + 5k$ che sono

DRLNSWWUI FLSRVAUSG JVYVWAAFT NGJWYEWDM MGUFVFDGK AAAJAJUDS LDSWSWSKK EAVYMYSUE UWGAWESWD GWEDLLKG

dall'analisi delle frequenze risulta che il cambio è $S \to a$.

Procedendo analogamente per le posizioni $2 + 5k$ $3 + 5k$, $4 + 5k$, $5k$ si trova che il testo in chiaro è

lamez zanot tedel venti april emill eotto cento quara ntase tteun acqua zzone diluv ialea ccomp agnat odasc rosci difol goree daimp etuos isoff idiv entos ubiss avala solit ariae selva ggiam ompra cemis olasi tuata sulle coste occid ental idibo rneoe ilcui nomeb astav ainqu eitem piasp arger eilte rrore acent olegh ealli ntorn olabi tazio nedel latig redel lamal esiap ostac omeaq uilas udiun agran rupet aglia taapi ccosu lmare acinq uecen topas sidal leult imeca panne delvi llagg iodig jehaw emque llano tteco ntroi lsoli toera

cioè

La mezzanotte del 20 aprile 1847, un acquazzone diluviale, accompagnato da scrosci di folgore e da impetuosi soffi di vento subissava la solitaria e selvaggia Mompracem, isola situata sulle coste occidentali di Borneo, e il cui nome bastava in quei tempi a spargere il terrore a cento leghe all'intorno. L'abitazione della Tigre della Malesia, posta come aquila su di una gran rupe tagliata a picco sul mare, a cinquecento passi dalle ultime capanne del villaggio di Gjehawem, quella notte, contro il solito, era

da *La tigre della Malesia* di Emilio Salgari e la chiave è **SLGRI**.

2.5 Crittoanalisi del cifrario di Hill

Il cifrario di Hill può essere difficile da violare con la sola conoscenza del testo cifrato, ma è più semplice se si conosce una porzione del testo in chiaro corrispondente.

Si supponga che si conosca il valore di m che è l'ordine possibile delle matrici che rappresentano la chiave del cifrario. Si supponga inoltre che si abbiano almeno m parti di testo in chiaro e del corrispondente testo cifrato, cioè

$$x_j = x_{j,1}, x_{j,2} \cdots x_{j,m}$$
$$y_j = y_{j,1}, y_{j,2} \cdots y_{j,m}$$

dove $y_j = e_K(x_j)$ e $1 \leq j \leq m$.

Siano $X = (x_{i,j})$ e $Y = (y_{i,j})$ delle matrici quadrate di ordine m e sia K la chiave che si intende determinare. Poiché vale che $Y = XK$, se X è invertibile, allora $K = X^{-1}Y$.

Per cui un esempio può essere $m = 2$ e che il testo in chiaro **friday** sia cifrato con **PQCFKU**. Quindi si ha che

$$e_k \begin{pmatrix} 5 \\ 17 \end{pmatrix} = \begin{pmatrix} 15 \\ 16 \end{pmatrix}, \; e_k \begin{pmatrix} 8 \\ 3 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

allora

$$K = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \begin{pmatrix} 11 & 2 \\ 21 & 25 \end{pmatrix}.$$

3 Segretezza Perfetta

Nel 1949, Claude Shannon pubblicò sulla rivista ***"Bell Systems Technical Journal"*** un articolo dal titolo ***"Communication Theory of Secrecy Systems"***. Questo articolo ebbe una notevole influenza sullo studio della crittografia.

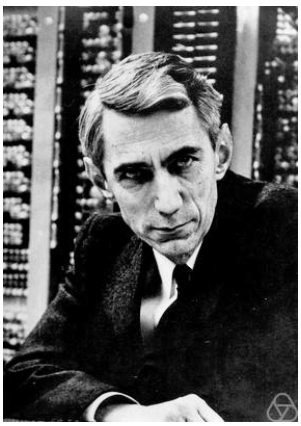


Figura 3.1: Claude Elwood Shannon (1916 – 2001) e il suo celebre articolo del 1949.

In questo capitolo discutiamo varie idee di Shannon. In primo luogo, tuttavia, consideriamo alcuni dei vari approcci per valutare la sicurezza di un crittosistema e definiamo alcuni dei criteri più utilizzati ora.

3.1 Criteri di Sicurezza

Definizione 3.1. Un crittosistema si dice **computazionalmente sicuro** se il miglior algoritmo per violare il sistema richiede almeno N operazioni, dove N è un numero specificato molto elevato.

Nella realtà nessun crittosistema è computazionalmente sicuro. Infatti, un crittosistema può essere computazionalmente sicuro **solo** rispetto a certi tipi di attacchi ma non ad altri.

Definizione 3.2. Un crittosistema si dice **dimostrabilmente sicuro** se la sua sicurezza si basa sulla difficoltà di risoluzione di un problema ampiamente studiato.

La sicurezza dimostrabile di un crittosistema **relativa** ad un problema non è una dimostrazione assoluta di sicurezza.

Definizione 3.3. Un crittosistema si dice **incondizionatamente sicuro** se esso non può essere violato anche se l'avversario è munito di risorse computazionali infinite.

Quando si studia la sicurezza di un crittosistema è necessario specificare il tipo di attacco rispetto al quale la si analizza. Per esempio, i crittosistemi affini e di Vigenère **non** sono computazionalmente sicuri rispetto all'attacco basato sulla conoscenza del testo cifrato, se l'avversario ha a disposizione una quantità sufficiente di quest'ultimo.

Analizziamo alcuni crittosistemi che sono incondizionatamente sicuri rispetto all'attacco basato sulla conoscenza del testo cifrato, se quest'ultimo è sufficientemente piccolo. Come conseguenza di quest'analisi, otterremo che i crittosistemi affini e di Vigenère sono incondizionatamente sicuri, se ogni messaggio in chiaro è cifrato con una chiave diversa, e nel caso di Vigenère, se il messaggio e la chiave hanno la stessa lunghezza.

3.2 Segretezza Perfetta

Sia $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un crittosistema in cui ogni chiave $K \in \mathcal{K}$ viene utilizzata per una sola cifratura. Supponiamo che

(1) Sia definita una distribuzione di probabilità nello spazio delle unità di messaggio in chiaro $\mathcal{P}$. Questa definisce una variabile aleatoria che denoteremo con $\mathbf{x}$. Quindi, la probabilità che la variabile $\mathbf{x}$ assuma valore x sarà denotata con $\mathbf{P}[\mathbf{x} = x]$.
(2) Sia definita una distribuzione di probabilità nello spazio delle chiavi $\mathcal{K}$. Questa definisce una variabile aleatoria che denoteremo con $\mathbf{K}$. La probabilità che la variabile $\mathbf{K}$ assuma valore K sarà denotata con $\mathbf{P}[\mathbf{K} = K]$.
(3) Le variabili $\mathbf{x}$ e $\mathbf{K}$ siano indipendenti (segue dal fatto che le chiavi vengono scelte dagli utenti crittosistema prima del testo da cifrare).

Le distribuzioni di probabilità su $\mathcal{P}$ e $\mathcal{K}$ inducono una distribuzione di probabilità su $\mathcal{C}$. Questa, a sua volta, definisce una variabile aleatoria $\mathbf{y}$.
Quindi

$$\begin{aligned}\mathbf{P}[\mathbf{y} = y] &= \sum_{\{K:y\in e_K(\mathcal{P})\}} \mathbf{P}[\mathbf{K} = K, \mathbf{x} = d_K(y)] = \\ &= \sum_{\{K:y\in e_K(\mathcal{P})\}} \mathbf{P}[\mathbf{K} = K]\mathbf{P}[\mathbf{x} = d_K(y)].\end{aligned}$$

Inoltre

$$\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x] = \sum_{\{K:x=d_K(y)\}} \mathbf{P}[\mathbf{K} = K].$$

E quindi, utilizzando il **Teorema di Bayes**, si ha che

$$\begin{aligned}\mathbf{P}[\mathbf{x} = x \mid \mathbf{y} = y] &= \frac{\mathbf{P}[\mathbf{x} = x]\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x]}{\mathbf{P}[\mathbf{y} = y]} = \\ &= \frac{\mathbf{P}[\mathbf{x} = x] \times \sum_{\{K:x=d_K(y)\}} \mathbf{P}[\mathbf{K} = K]}{\sum_{\{K:y\in e_K(\mathcal{P})\}} \mathbf{P}[\mathbf{K} = K]\mathbf{P}[\mathbf{x} = d_K(y)]}.\end{aligned}$$

Esempio 3.4. Si consideri il crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, dove $\mathcal{P} = \{a, b\}$, $\mathcal{C} = \{1, 2, 3, 4\}$ e $\mathcal{K} = \{K_1, K_2, K_3\}$ e dove $\mathcal{E}, \mathcal{D}$ sono definite mediante la seguente **matrice di cifratura**

	a	b
K_1	1	2
K_2	2	3
K_3	3	4

e supponiamo che soddisfi le ipotesi **(1)**–**(3)**.
Quindi, siano $\mathbf{P}[a] = 1/4$ e $\mathbf{P}[b] = 3/4$, e $\mathbf{P}[K_1] = 1/2$ e $\mathbf{P}[K_2] = \mathbf{P}[K_3] = 1/4$ le distribuzioni di probabilità su $\mathcal{P}$ e $\mathcal{K}$, rispettivamente.

Calcoliamo la distribuzione di probabilità su $\mathcal{C}$

$$\begin{aligned}\mathbf{P}[1] &= \mathbf{P}[K_1]\mathbf{P}[a] = 1/8 \\ \mathbf{P}[2] &= \mathbf{P}[K_1]\mathbf{P}[b] + \mathbf{P}[K_2]\mathbf{P}[a] = 3/8 + 1/16 = 7/16 \\ \mathbf{P}[3] &= \mathbf{P}[K_2]\mathbf{P}[b] + \mathbf{P}[K_3]\mathbf{P}[a] = 3/16 + 1/16 = 1/4 \\ \mathbf{P}[4] &= \mathbf{P}[K_3]\mathbf{P}[b] = 3/16.\end{aligned}$$

Infine, calcoliamo la probabilità condizionata

$$\begin{aligned}
\mathbf{P}[a \mid 1] &= \frac{\mathbf{P}[a]\mathbf{P}[K_1]}{\mathbf{P}[1]} = \frac{1/4 \times 1/2}{1/8} = 1 \\
\mathbf{P}[a \mid 2] &= \frac{\mathbf{P}[a]\mathbf{P}[K_2]}{\mathbf{P}[2]} = \frac{1/4 \times 1/4}{7/16} = 1/7 \\
\mathbf{P}[a \mid 3] &= \frac{\mathbf{P}[a]\mathbf{P}[K_3]}{\mathbf{P}[3]} = \frac{1/4 \times 1/4}{1/4} = 1/4 \\
\mathbf{P}[a \mid 4] &= \frac{\mathbf{P}[a] \times 0}{\mathbf{P}[4]} = 0.
\end{aligned}$$

In modo analogo si calcolano $\mathbf{P}[b \mid y]$ con $y \in \mathcal{C}$.

Definizione 3.5. Un crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ fornisce **segretezza perfetta** se per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ risulta $\mathbf{P}[x \mid y] = \mathbf{P}[x]$.

In altre parole, i crittosistemi dotati di segretezza perfetta sono tutti e soli quelli in cui la conoscenza del testo cifrato non fornisce alcuna informazione sul testo in chiaro.

Lemma 3.6. *Se un crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ fornisce segretezza perfetta, allora per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ esiste almeno una chiave $K \in \mathcal{K}$ tale che $y = e_K(x)$. In particolare, $|\mathcal{K}| \geq |\mathcal{P}|$.*

Dimostrazione. Siano $x \in \mathcal{P}$ e $y \in \mathcal{C}$, allora per il **Teorema di Bayes** e per la segretezza perfetta vale che

$$\mathbf{P}[y \mid x] = \mathbf{P}[y].$$

Possiamo assumere che $\mathbf{P}[y] > 0$ per ogni $y \in \mathcal{C}$, altrimenti consideriamo $(\mathcal{P}, \mathcal{C}', \mathcal{K}, \mathcal{E}, \mathcal{D})$ con $\mathcal{C}' = \mathcal{C} - \{y\}$. Pertanto, per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ vale che $\mathbf{P}[y \mid x] = \mathbf{P}[y] > 0$. Quindi, esiste almeno una chiave $K \in \mathcal{K}$ tale che $y = e_K(x)$. Pertanto, fissato $x \in \mathcal{P}$, vale che

$$\mathcal{C} = \{e_K(x) : K \in \mathcal{K}\}.$$

Da ciò segue che $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{P}|$.

□

3.3 Teorema di Shannon

Forniamo ora una caratterizzazione della perfetta sicurezza, dovuta originariamente a Shannon (1949), per i crittosistemi in cui vale $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$.

Teorema 3.7. *Un crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, tale che $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$, fornisce segretezza perfetta se, e solo se, valgono i seguenti fatti:*

1. *Le chiavi sono equiprobabili con probabilità $1/|\mathcal{K}|$;*
2. *Per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ esiste un unica una chiave $K \in \mathcal{K}$ tale che $y = e_K(x)$.*

Dimostrazione. Supponiamo che valgano **(1)** e **(2)**. Allora segue che

$$\begin{aligned}
\mathbf{P}[\mathbf{y} = y] &= \sum_{\{K : y \in e_K(\mathcal{P})\}} \mathbf{P}[\mathbf{K} = K]\mathbf{P}[\mathbf{x} = d_K(y)] = \\
&= \sum_{K \in \mathcal{K}} \frac{1}{|\mathcal{K}|}\mathbf{P}[\mathbf{x} = d_K(y)] = \\
&= \frac{1}{|\mathcal{K}|} \sum_{K \in \mathcal{K}} \mathbf{P}[\mathbf{x} = d_K(y)].
\end{aligned}$$

Per **(2)** e per $|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$ segue che $\{d_K(y) : K \in \mathcal{K}\} = \mathcal{P}$. Quindi,

$$\sum_{K \in \mathcal{K}} \mathbf{P}[\mathbf{x} = d_K(y)] = \sum_{x \in \mathcal{P}} \mathbf{P}[\mathbf{x} = x] = 1$$

e quindi $\mathbf{P}[\mathbf{y} = y] = 1/|\mathcal{K}|$. D'altra parte, **(1)** e **(2)** implicano

$$\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x] = \mathbf{P}[\mathbf{K} = K] = \frac{1}{|\mathcal{K}|},$$

dove K è la chiave tale che $y = e_K(x)$. Pertanto,

$$\begin{aligned}
\mathbf{P}[\mathbf{x} = x \mid \mathbf{y} = y] &= \frac{\mathbf{P}[\mathbf{x} = x]\mathbf{P}[\mathbf{y} = y, \mathbf{x} = x]}{\mathbf{P}[\mathbf{y} = y]} = \\
&= \frac{\mathbf{P}[\mathbf{x} = x] \times \frac{1}{|\mathcal{K}|}}{\frac{1}{|\mathcal{K}|}} = \\
&= \mathbf{P}[\mathbf{x} = x]
\end{aligned}$$

che è la segretezza perfetta.

Supponiamo che il crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ fornisca segretezza perfetta. Fissato $x \in \mathcal{P}$, dal **Lemma 3.6** segue che per ogni $y \in \mathcal{C}$ esiste almeno un $K \in \mathcal{K}$ tale che $y = e_K(x)$. Allora

$$|\mathcal{C}| = |\{e_K(x) : K \in \mathcal{K}\}| \leq |\mathcal{K}|$$

e quindi $|\{e_K(x) : K \in \mathcal{K}\}| = |\mathcal{K}|$, essendo $|\mathcal{C}| = |\mathcal{K}|$.
Pertanto, per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ esiste un unica una chiave $K \in \mathcal{K}$ tale che $y = e_K(x)$. Abbiamo così provato che vale **(2)**.

Sia $n = |\mathcal{K}|$, allora $\mathcal{P} = \{x_i : 1 \leq i \leq n\}$. Fissato $y \in \mathcal{C}$, indicizziamo gli elementi di $\mathcal{K}$ in modo tale che per ogni $1 \leq i \leq n$ risulti $e_{K_i}(x_i) = y$. Sfruttando l'ipotesi di segretezza perfetta e il **Teorema di Bayes** segue che

$$\mathbf{P}[\mathbf{x} = x_i] = \mathbf{P}[\mathbf{x} = x_i \mid \mathbf{y} = y] = \frac{\mathbf{P}[\mathbf{x} = x_i]\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x_i]}{\mathbf{P}[\mathbf{y} = y]}.$$

Quindi

$$\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x_i] = \mathbf{P}[\mathbf{y} = y].$$

Siccome K_i è l'unica chiave tale che $e_{K_i}(x_i) = y$, si ha

$$\mathbf{P}[\mathbf{y} = y \mid \mathbf{x} = x_i] = \mathbf{P}[\mathbf{K} = K_i].$$

Pertanto

$$\mathbf{P}[\mathbf{K} = K_i] = \mathbf{P}[\mathbf{y} = y].$$

Quindi le chiavi sono tutte equiprobabili con probabilità $\mathbf{P}[\mathbf{y} = y]$ con y fissato. Siccome le chiavi sono $|\mathcal{K}|$, segue **(1)**.

□

Corollario 3.8. *Se le* 26 *chiavi nel cifrario mediante spostamento sono equiprobabili, allora il cifrario mediante spostamento fornisce segretezza perfetta indipendentemente dalla distribuzione di probabilità del testo in chiaro.*

Dimostrazione. Segue dal **Teorema 3.7**, basta notare che $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$ e che per ogni $x, y \in \mathbb{Z}_{26}$ vale che $K = y - x$ è l'unica chiave che realizza $y = e_K(x)$.

□

Corollario 3.9. *Se le* 26^m *chiavi nel cifrario di Vigènere sono equiprobabili e ogni chiave è utilizzata per cifrare una sola unità di messaggio in chiaro, allora il cifrario fornisce segretezza perfetta indipendentemente dalla distribuzione di probabilità del testo in chiaro.*

Dimostrazione. Segue dal **Teorema 3.7**, per le ipotesi fatte, siccome

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m.$$

□

Una ben nota realizzazione della segretezza perfetta è il cifrario One-time Pad, che fu costruito da Vernam nel 1917 nell'ambito della cifratura e decifratura dei messaggi telegrafici.

Figura 3.2: Gilbert Sandford Vernam (1890 – 1960)

Definizione 3.10. (Cifrario One-time Pad).
Sia $n \geq 1$ un intero, allora il crittosistema $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ dove $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_2)^n$ e dove, fissato $K \in \mathcal{K}$, per ogni $x \in \mathcal{P}$ e $y \in \mathcal{C}$ vale che $e_K(x) = x \oplus K$ e $d_K(y) = y \oplus K$.

Se le chiavi nel cifrario One-time pad sono equiprobabili e ogni chiave è utilizzata per cifrare una sola unità di messaggio in chiaro, allora il cifrario fornisce segretezza perfetta per il **Teorema 3.7**.

Limitazioni dei crittosistemi incondizionatamente sicuri:

- Poiché $|\mathcal{K}| \geq |\mathcal{P}|$, una notevole quantità di chiavi deve essere trasmessa attraverso un canale sicuro. Questa limitazione sarebbe relativa, se una chiave potesse essere utilizzata per cifrare più messaggi. Tuttavia la perfetta segretezza depende dal fatto che ogni chiave è utilizzata per una sola cifratura. Questo crea grossi problemi con la gestione delle chiavi.
- Nel caso del cifrario One-time pad è fondamentale che ogni chiave sia utilizzata per una sola cifratura, altrimenti sarebbe il cifrario sarebbe vulnerabile all'attacco basato sulla conoscenza del testo in chiaro. Infatti, $K = x \oplus e_K(x)$.

3.4 Cifrari Prodotto

Nel 1949 il noto matematico Claude Shannon nel suo **"Communication Theory of Secrecy Systems"**, propose di combinare due o più crittosistemi, attraverso il loro **prodotto**. Quest'operazione determina quindi un terzo sistema partendo da due crittosistemi dati. L'idea dei cifrari prodotto avrà un ruolo fondamentale nella creazione di alcuni tra i più moderni cifrari, come ad esempio l'**Advanced Encryption Standard**.

Definiamo ora nel dettaglio il crittosistema prodotto.

Definizione 3.11. (Crittosistema Prodotto).
Siano $\mathbb{S}_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$ e $\mathbb{S}_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$, allora il prodotto di $\mathbb{S}_1$ e $\mathbb{S}_2$ è il crittosistema

$$\mathbb{S}_1 \times \mathbb{S}_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D}),$$

in cui le funzioni di cifratura e decifratura sono

$$e_{(K_1,K_2)} = e_{K_1} \circ e_{K_2}$$
$$d_{(K_1,K_2)} = d_{K_2} \circ d_{K_1},$$

dove (e_{K_1}, d_{K_1}), (e_{K_2}, d_{K_2}) sono le funzioni di cifratura e decifratura di $\mathbb{S}_1$ e $\mathbb{S}_2$, rispettivamente.

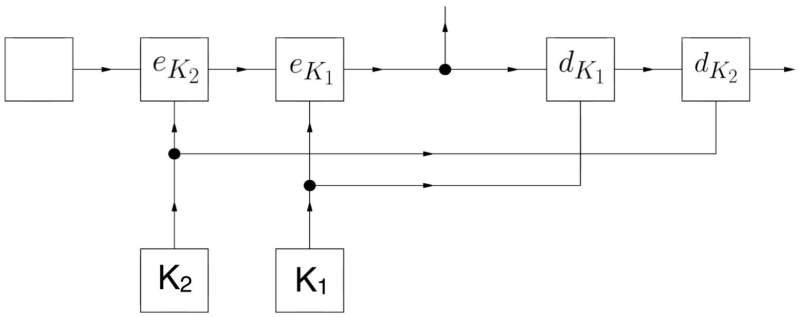


Figura 3.3: Prodotto di due sistemi $\mathbb{S}_1, \mathbb{S}_2$

Chiaramente, vale che $\mathbb{S}_1 \times \mathbb{S}_2 \neq \mathbb{S}_2 \times \mathbb{S}_1$. Qualora si verifichi che $\mathbb{S}_1 \times \mathbb{S}_2 = \mathbb{S}_2 \times \mathbb{S}_1$ allora si dice che $\mathbb{S}_1$ e $\mathbb{S}_2$ **commutano**.

Proponiamo ora un esempio di cifrario prodotto.
Sia $\mathbb{S}_1$ il cifrario moltiplicativo, quindi $e_a : x \mapsto ax \ mod\ 26$ con $\mathrm{mcd}(a, 26) = 1$, e $\mathbb{S}_2$ il cifrario mediante spostamento $e_b : x \mapsto x + b \ mod\ 26$. Allora il cifrario prodotto $\mathbb{S}_1 \times \mathbb{S}_2$ ha come insieme delle chiavi $\mathcal{K} = \{(a, b) : \mathrm{mcd}(a, 26) = 1\}$, e come funzione di cifratura e decifratura

$$e_{(a,b)} : x \mapsto ax + b \ mod\ 26 \qquad d_{(a,b)} : y \mapsto a^{-1}(y - b) \ mod\ 26.$$

Quindi, il cifrario prodotto del cifrario moltiplicativo e di quello mediante spostamento è affine. È facile vedere, inoltre, che il cifrario moltiplicativo e quello mediante spostamento commutano.

Un crittosistema endomorfo $\mathbb{S}$ si definisce **idempotente** se $\mathbb{S}^2 = \mathbb{S}$. In generale, la sicurezza del crittosistema $\mathbb{S}^n$ è incrementata solo quando **non** è idempotente.

Proposizione 3.12. Il cifrario di Vigenére è idempotente.

Dimostrazione. Sia $\mathbb{S}$ un cifrario di Vigenére, (K, K') una chiave di $\mathbb{S}^2$ e $(x_1, \ldots, x_n)$ una n-upla di un'unità di messaggio in chiaro.
Segue dalla definizione di cifrario prodotto che

$$\begin{aligned} e_{(K,K')}(x_1, \ldots, x_m) &= (e_{K'} \circ e_K)(x_1, \ldots, x_m) \\ &= e_{K'}(e_K(x_1, \ldots, x_m)) \\ &= e_{K'}(x_1 + k_1, \ldots, x_m + k_m) \\ &= ((x_1 + k_1) + k'_1, \ldots, (x_m + k_m) + k'_m) \\ &= (x_1 + (k_1 + k'_1), \ldots, x_m + (k_m + k'_m)) \\ &= e_{(K+K')}(x_1, \ldots, x_m). \end{aligned}$$

Pertanto $e_{(K,K')} = e_{K+K'}$ e in modo analogo si prova che $d_{(K,K')} = d_{K+K'}$. Quindi $\mathbb{S}^2 = \mathbb{S}$, che è l'asserto.

□

Lemma 3.13. *Siano $\mathbb{S}_1, \mathbb{S}_2$ due crittosistemi. Se $\mathbb{S}_1$ e $\mathbb{S}_2$ sono idempotenti e commutano, allora il loro prodotto è idempotente.*

Dimostrazione. Per dimostrarlo si sfrutta la proprietà associativa del prodotto. Quindi,

$$\begin{aligned} (\mathbb{S}_1 \times \mathbb{S}_2)^2 &= (\mathbb{S}_1 \times \mathbb{S}_2) \times (\mathbb{S}_1 \times \mathbb{S}_2) \\ &= \mathbb{S}_1 \times (\mathbb{S}_2 \times \mathbb{S}_1) \times \mathbb{S}_2 \\ &= (\mathbb{S}_1 \times \mathbb{S}_1) \times (\mathbb{S}_2 \times \mathbb{S}_2) \\ &= \mathbb{S}_1 \times \mathbb{S}_2. \end{aligned}$$

□

Un modo per costruire crittosistemi non idempotenti è quello di considerare il prodotto di crittosistemi semplici, eventualmente idempotenti, che non commutino.

4 Advanced Encryption Standard

4.1 Cifrari Iterati

Analizziamo ora una particolare classe di cifrari a blocchi: i **cifrari iterati**. Per costruire un cifrario iterato è necessario specificare una **funzione round** e una **programmazione delle chiavi**. La cifratura di un testo in chiaro avviene attraverso un numero finito N_r di round (o iterazioni). Fissata una chiave arbitraria K, rappresentata da una stringa binaria di lunghezza fissata, attraverso un algoritmo pubblico viene generata una sequenza $(K^1, \ldots, K^{N_r})$ di sottochiavi dette **chiavi round**.

La **funzione round** g ha due input: lo **stato** che denotiamo con w^{r-1} e la **chiave round**. Lo stato successivo è definito come $w^r = g(w^{r-1}, K^r)$.

Lo stato iniziale w^0 è il testo in chiaro x, il testo cifrato y corrispondente ad x è lo stato dopo che tutte le N_r iterazioni sono state eseguite. La cifratura avviene come segue:

$$\begin{aligned} w^0 &\leftarrow x \\ w^1 &\leftarrow g(w^0, K^1) \\ w^2 &\leftarrow g(w^1, K^2) \\ &\vdots \\ w^{N_r-1} &\leftarrow g(w^{N_r-2}, K^{N_r-1}) \\ w^{N_r} &\leftarrow g(w^{N_r-1}, K^{N_r}) \\ y &\leftarrow w^{N_r}. \end{aligned}$$

Affinchè la decifratura sia possibile, la funzione deve essere iniettiva rispetto alla prima variabile una volta che la seconda è fissata. Ciò equivale a dire che esiste una funzione g^{-1} con la proprietà $g^{-1}(g(w, y), y) = w$ per tutti i w e y.

La decifratura avviene come segue:

$$\begin{aligned} w^{N_r} &\leftarrow y \\ w^{N_r-1} &\leftarrow g^{-1}(w^{N_r}, K^{N_r}) \\ &\vdots \\ w^1 &\leftarrow g^{-1}(w^2, K^2) \\ w^0 &\leftarrow g^{-1}(w^1, K^1) \\ x &\leftarrow w^0. \end{aligned}$$

4.2 Substitution-Permutation Network

Nei cifrari appena presentati, le funzioni di cifratura e_K sono legate unicamente alle chiavi, perciò queste ultime sono variabili.
In seguito, analizzeremo invece crittosistemi iterati, tali che alcuni dei singoli cifrari presenti al loro interno risultano essere **fissati**, e pertanto non dipendenti più dalle chiavi. Il primo ad essere proposto è il **Substitution-Permutation Network (SPN)**.
Poichè si tratta di un crittosistema moderno, e quindi utilizzato principalmente da calcolatori (che lavorano in binario), l'alfabeto è $\mathbb{Z}_2$.
Analizziamo ora, in maniera pù dettagliata, la struttura del crittosistema SPN.
L'insieme dei messaggi in chiaro così come quelli cifrati è l'insieme delle stringhe binarie di lunghezza ℓm, dove ℓ ed m sono interi positivi fissati.
L'insieme delle chiavi è

$$\mathcal{K} = \{(K^1, \dots, K^{N_r+1})\} : K^i \in \{0,1\}^{\ell m}, 1 \leq i \leq N_r + 1\}.$$

Una qualsiasi chiave $(K^1, \dots, K^{N_r+1})$ è derivata da una chiave iniziale K, secondo un opportuno algoritmo pubblico.
La funzione di cifratura si basa su due permutazioni $\pi_S \in Sym(\{0,1\}^\ell)$ detta **S-box** (scatola di sostituzione) e $\pi_P \in Sym(\ell m)$.

Sia $x = (x_1, \dots, x_{\ell m})$ un messaggio in chiaro, allora x può essere riguardata come la concatenazione di m sottostringhe binarie denotate con $x_{<i>}$. Quindi

$$x = x_{<1>} || \dots || x_{<m>},$$

dove $x_{<i>} = (x_{(i-1)\ell+1}, \dots, x_{i\ell})$ per ogni $\leq i \leq m$.
La cifratura di x avviene secondo il seguente algoritmo:

Algoritmo 4.1. **(SPN** $(x, \pi_S, \pi_P, (K^1, ..., K^{N_r+1}))$**)**

$w^0 \leftarrow x$
for $r \leftarrow 1$ **to** $N_r - 1$
do
- $u^r \leftarrow w^{r-1} \oplus K^r$
- **for** $i \leftarrow 1$ **to** m
  - **do** $v^r_{(i)} \leftarrow \pi_S(u^r_{(i)})$
- $w^r \leftarrow (v^r_{\pi_{P(1)}}, ..., v^r_{\pi_{P(\ell m)}})$

$u^{N_r} \leftarrow w^{N_r-1} \oplus K^{N_r}$
for $i \leftarrow 1$ **to** m
do $v^{N_r}_{(i)} \leftarrow \pi_S(u^{N_r}_{(i)})$
$y \leftarrow v^{N_r} \oplus K^{N_r+1}$
output (y)

Come si evince dall'**Algoritmo 4.1**, l'SPN consiste di N_r iterazioni. In ciascuna di queste, tranne l'ultima, sulle m sottostringhe viene applicata prima una permutazione π_S e successivamente queste vengono permutate attraverso π_P.
Nella prima iterazione si calcola $u^1 = x \oplus K^1$ (questo procedimento è detto **sbiancamento**). Successivamente viene applicata π_S alle m sottostringhe di $u^1 = u^1_{<1>} \| \dots \| u^1_{<m>}$ ottenendo così $v^1 = v^1_{<1>} \| \dots \| v^1_{<m>}$ ($v^1_{<i>} = \pi_S(u^1_{<i>})$).
La prima iterazione termina con l'applicazione di π_P sulle ℓm posizioni di v^1 generando così $w^1 = v^1_{\pi_P(1)} \cdots v^1_{\pi_P(m)}$.
La seconda iterazione è analoga alla prima con $u^2 = w^1 \oplus K^2$ al posto di $u^1 = x \oplus K^1$, dove w^1 è la stringa generata nella prima iterazione.
Dopo $N_r - 1$ iterazioni viene così generato $u^{N_r} = w^{N_r - 1} \oplus K^{N_r}$. Applicando π_S a u^{N_r}, si ha $v^{N_r} = \pi_S(u^{N_r}_{<1>}) \| \dots \| \pi_S(u^{N_r}_{<m>})$. Infine l'ultima operazione dell'algoritmo è $y = v^{N_r} \oplus K^{N_r+1}$ (**sbiancamento**).

L'SPN ha diverse caratteristiche interessanti, il design è semplice ed estremamente efficiente sia nell'hardware che nel software. Per quanto riguarda il software, π_S viene implementata generalmente come una tabella dati (risulta più veloce consultare la tabella piuttosto che calcolare ogni volta il risultato di π_S applicato ad una stringa). Poichè $\pi_S \in Sym(\{0,1\}^\ell)$, la memoria da immagazzinare è di circa $\ell \cdot 2^\ell$ bit perchè bisogna immagazzinare 2^ℓ stringhe binarie ognuna di lunghezza ℓ. L'implementazione hardware, invece, necessita di avere π_S relativamente piccolo.
Di seguito viene riportata una versione esemplificata del crittosistema SPN in cui $\ell = m = N_r = 4$. Quindi, $\mathcal{P} = \mathcal{C} = \{0,1\}^{16}$. Le permutazioni $\pi_P \in Sym(16)$ e $\pi_S \in Sym(\{0,1\}^4)$ sono definite come segue:

$$\pi_P = (2\ 5)(3\ 9)(4\ 13)(7\ 10)(8\ 14)(12\ 15).$$

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

Tabella 4: S-Box.

Per completare l'esempio, è necessario definire un algoritmo che generi la sequenza delle sottochiavi a partire da una chiave iniziale K. Per fare ciò, un possibile algoritmo è quello che da come output la sequenza $(K^1, K^2, K^3, K^4, K^5)$ (si ricordi che $N_r = 4$), dove

$$K^i = (k_{4i-3}, k_{4i-2}, \ldots, k_{4i+12}).$$

Sia $K = 11010010000101111011101000111100$. Quindi le 5 sottochiavi sono

$$\begin{aligned} K^1 &= 1101\,0010\,0001\,0111 \\ K^2 &= 0010\,0001\,0111\,1011 \\ K^3 &= 0001\,0111\,1011\,1010 \\ K^4 &= 0111\,1011\,1010\,0011 \\ K^5 &= 1011\,1010\,0011\,1100 \end{aligned}$$

Supponiamo che il testo in chiaro da cifrare sia:

$$\mathbf{x{=}1001111000010001}.$$

Allora la cifratura di x procede come segue:

$$\begin{aligned} w^0 &= (1001)(1110)\ (0001)(0001) && \oplus \\ K^1 &= (1101)(0010)\ (0001)(0111) && = \\ u^1 &= (0100)(1100)\ (0000)(0110) && \swarrow \pi_S \\ v^1 &= (0010)(0101)\ (1110)(1011) && \swarrow \pi_P \\ w^1 &= (0011)(0110)\ (1011)(0101) && \oplus \\ K^2 &= (0010)(0001)\ (0111)(1011) && = \\ u^2 &= (0001)(0111)\ (1100)(1110) && \swarrow \pi_S \\ v^2 &= (0100)(1000)\ (0101)(0000) && \swarrow \pi_P \\ w^2 &= (0100)(1010)\ (0000)(0010) && \oplus \\ K^3 &= (0001)(0111)\ (1011)(1010) && = \\ u^3 &= (0101)(1101)\ (1011)(1000) && \swarrow \pi_S \\ v^3 &= (1111)(1001)\ (1100)(0011) && \swarrow \pi_P \\ w^3 &= (1110)(1010)\ (1001)(1101) && \oplus \\ K^4 &= (0111)(1011)\ (1010)(0011) && = \\ u^4 &= (1001)(0001)\ (0011)(1110) && \swarrow \pi_S \\ v^4 &= (1010)(0100)\ (0001)(0000) && \oplus \\ K^5 &= (1011)(1010)\ (0011)(1100) && = \\ y &= (0001)(1010)\ (0010)(1100) \end{aligned}$$

Pertanto **x=1001111000010001** viene cifrato con **y=0001101000101100**.

Dall'analisi della struttura del SPN si evince che sebbene le operazioni di sostituzione e di permutazione siano indipendenti dalla chiave, mentre l'iterazione non lo è.

Nell'esempio che abbiamo utilizzato, ad ogni iterazione la memoria richiesta era di 2^6 bit ($\ell = 4$). Si noti che per $\ell = 16$ la memoria occupata crescerebbe fino a 2^{20} diminuendo così l'efficienza del crittosistema. Lo scopo dell'esempio utilizzato è puramente illustrativo poichè la chiave iniziale consiste di 32 bit e quindi il sistema può essere violato attraverso la forza bruta.

Pertanto, nella realtà, per ottenere cifrari SPN sicuri, bisogna aumentare la lunghezza della chiave, la grandezza degli **S-box** e inoltre un maggior numero di iterazioni.

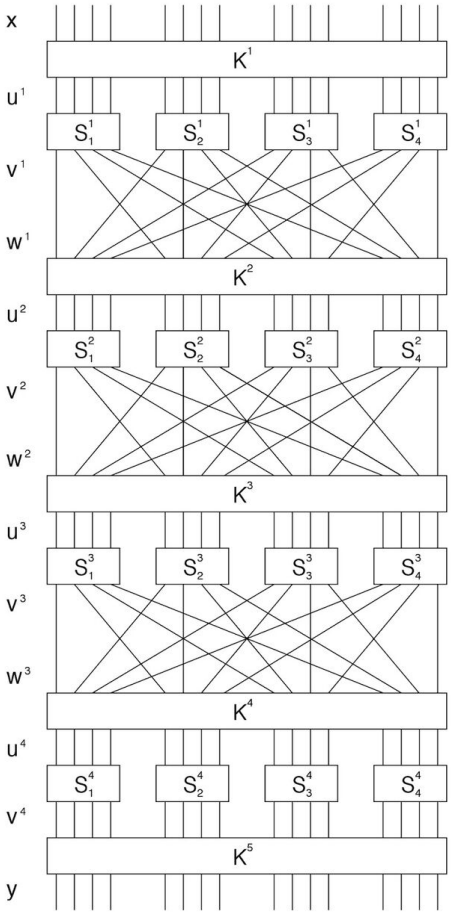


Figura 4.1: Struttura di un cifrario SPN

4.3 Advanced Encryption Standard

Nel 1997 il National Insitute of Standards and Technology (NIST) istituì una competizione per la creazione di un **crittosistema simmetrico** per proteggere le informazioni federali sensibili.

I cifrari proposti dai candidati dovevano assicurare requisiti minimi che erano:

- **Lunghezza del blocco** pari a 128 bit.
- **Lunghezza della chiave iniziale** di 128, 192 o 256 bit.

I criteri di valutazione erano i seguenti:

- **Sicurezza**: È il criterio principale di valutazione. Esso comprende resistenza dell'algoritmo alla crittanalisi, solidità delle basi matematiche, casualità dell'output dell'algoritmo e maggiore sicurezza rispetto ad altri candidati.
- **Costo**: Altro importante criterio, riguarda l'efficienza computazionale (velocità) su varie piattaforme, requisiti di licenza e di memoria.
- **Caratteristiche algoritmiche e di implementazione**: Flessibilità, semplicità algoritmica e adattabilità software e hardware.

La competizione fu vinta nel 2001 da due ricercatori belgi Vincent Rijmen e Joan Daemen con il crittosistema da essi inventato detto **Rijndael** (dalle loro iniziali) che successivamente venne denominato **Advanced Encryption Standard (AES)**.

Figura 4.2: Vincent Rijmen (1970) e Joan Daemen (1965)

4.3.1 Conversione Binario-Esadecimale-Elemento di $GF(2^8)$

Le unità di messaggio in chiaro e cifrato del cifrario AES sono costituite da 16 byte. Per una migliore comprensione delle operazioni utilizzate dal suddetto cifrario è utile convertire i byte sia in coppie di cifre esadecimali sia come elementi del campo finito $GF(2^8)$. La conversione da binario a esadecimale è fornita della seguente tabella.

BINARIO	ESA	BINARIO	ESA
(0000)	0	(1000)	8
(0001)	1	(1001)	9
(0010)	2	(1010)	A
(0011)	3	(1011)	B
(0100)	4	(1100)	C
(0101)	5	(1101)	D
(0110)	6	(1110)	E
(0111)	7	(1111)	F

Tabella 5: Conversione da binario a esadecimale.

Attraverso la **Tabella 5** è facile vedere che, ad esempio, la conversione esadecimale del byte 1001, 1110 è $9E$.

Prima di fornire la conversione dei byte in elementi di $GF(2^8)$, diamo una breve descrizione di quest'ultimo.

Si consideri il polinomio $p(x) = x^8 + x^4 + x^3 + x + 1$ a coefficienti in $GF(2)$. Poichè $p(x)$ è irriducibile su $GF(2)$, l'ideale $\langle p(x) \rangle$ è massimale in $GF(2)[x]$ e quindi $\frac{GF(2)[x]}{\langle p(x) \rangle}$ è una copia isomorfa del campo $GF(2^8)$.

Pertanto, gli elementi di $GF(2^8)$ sono descrivibili nella forma $\langle p(x) \rangle + a(x)$ con $a(x) \in GF(2)[x]$. Utilizzando la divisione tra polinomi è facile vedere che ogni laterale $\langle p(x) \rangle + a(x)$ ha un unico rappresentante di grado compreso tra 0 e 7, corrispondente al resto della divisione del polinomio $a(x)$ con $p(x)$.

Quindi,

$$GF(2^8) = \{b_7 x^7 + b_6 x^6 + \cdots + b_1 x + b_0,\ b_7, \ldots, b_0 \in GF(2)\}$$

in cui la somma tra due rappresentanti $b(x)$ e $b'(x)$ coincide con quella usuale in $GF(2)[x]$. Il prodotto $b(x)b'(x)$ è uguale a $r(x)$, dove $r(x)$ è il polinomio resto della divisione di $b(x)b'(x)$ per $p(x)$. In particolare, il rappresentante dell'inverso del polinomio $b(x)$ si calcola attraverso il ben noto algoritmo euclideo.

La conversione dei byte in elementi di $GF(2^8)$ avviene come segue.
Sia $b = b_7b_6b_5b_4b_3b_2b_1b_0$ il generico byte, ad esso viene associato il polinomio

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

e quindi l'elemento del campo $GF(2^8)$ individuato da quest'ultimo.
Pertanto, per esempio, alla stringa binaria 10011110 viene fatto corrispondere l'elemento del campo individuato dal polinomio $b(x) = x^7 + x^4 + x^3 + x^2 + x$.

Per il calcolo dell'inverso di $b(x)$ in $GF(2^8)$ si applica l'algoritmo euclideo:

1. $p(x) = b(x)x + (x^5 + x^2 + x + 1)$.
2. $b(x) = (x^5 + x^2 + x + 1)x^2 + x$.
3. $(x^5 + x^2 + x + 1) = x(x^4 + x + 1) + 1$.

Quindi,

$$1 = p(x)(x^6 + x^3 + x^2 + 1) + b(x)(x^7 + x^3 + 1).$$

Pertanto l'inverso di $b(x)$ *mod* $p(x)$ è $x^7 + x^3 + 1$, che corrisponde alla stringa binaria 10001001.

Per velocizzare la procedura di calcolo dell'inverso in $GF(2^8)$, il calcolatore si avvale della seguente tabella.

		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
	1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
	2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
	3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
	4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
	5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
	6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
X	7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
	8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
	9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
	A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
	B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
	C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
	D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
	E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
	F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

Tabella 6: Tabella degli inversi in $GF(2^8)$.

Infatti l'inverso di $b = 10011110 = 9E$ è dato dall'elemento nella riga 9 e colonna E della suddetta tabella che è 89 = 10001001 (che corrisponde a = $x^7 + x^3 + 1$).

Si noti che la procedura è coerente con il calcolo in $GF(2^8)$ effettuato prima. In realtà, il calcolatore utilizza l'equivalente binario della **Tabella 6**, bypassando la conversione in esadecimale e velocizzando ulteriormente la procedura per il calcolo degli inversi in $GF(2^8)$.

4.4 Struttura

Il cifrario AES è un particolare cifrario iterato in cui i messaggi in chiaro, così come quelli cifrati, sono stringhe binarie di lunghezza 128, ovvero blocchi binari di 4×4 byte.

Le chiavi hanno tre possibili lunghezze 128, 192 o 256 bit. Il numero N_r di iterazioni è 10, 12 o 14 a seconda che le chiavi abbiano lunghezza 128, 192 o 256 bit, rispettivamente. Le sottochiavi $(K_0, K_1, \ldots, K_{N_r})$ vengono calcolate attraverso un algoritmo pubblico che ha come input una chiave iniziale segreta K. La lunghezza di quest'ultima quindi determina sia il numero N_r di iterazioni che la lunghezza delle sottochiavi utilizzate in ognuna di esse. L'output di ogni iterazione è detto **stato**. Ogni iterazione, eccetto l'ultima, consiste di quattro operazioni:

1. **SubBytes**: Sostituzione attraverso l'S-box, ovvero si applica un opportuno elemento di $Sym\left(\{0,1\}^8\right)$.
2. **ShiftRows**: Permutazione sulle righe dello stato.
3. **MixColumns**: Trasformazione lineare sui blocchi di 4 byte.
4. **Addroundkey**: Operazione di XOR tra una sottochiave e lo stato.

Nell'ultima iterazione, l'operazione **MixColumn** non viene effettuata.

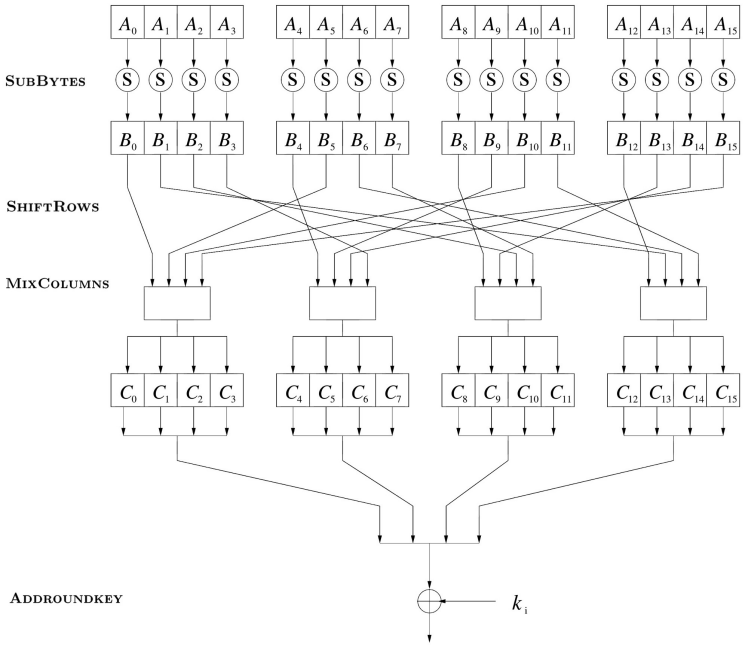


Figura 4.3: i-esima funzione round dell'AES, per $i = 1, \ldots, N_r - 1$

Esistono forti analogie tra la struttura del SPN e quella dell'AES: entrambi i crittosistemi presentano una combinazione di sottochiavi, una sostituzione ed una permutazione. Inoltre, contengono l'operazione di **sbiancamento**. La principale differenza tra questi due cifrari risiede nella presenza, per l'AES, della trasformazione lineare **MixColumn**.

Le variabili dell'AES, ovvero gli stati, sono rappresentate da una matrice 4×4 di byte:

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Lo stato iniziale w^0, che corrisponde al testo in chiaro $x = (x_0, x_1, \ldots, x_{15})$, è

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

Analizziamo i singoli livelli dell'AES[1].

4.4.1 SubBytes

L'operazione **SubBytes** è una particolare sostituzione che opera in modo indipendente su tutti i byte dello stato applicando una **S-box** su ognuno di essi. In particolare, si tratta di una permutazione di elementi di $\{0,1\}^8$ che opera come segue: fissata una generica stringa binaria non nulla $\alpha_7\alpha_6\alpha_5\alpha_4\alpha_3\alpha_2\alpha_1\alpha_0$ si determina l'inversa $a_7a_6a_5a_4a_3a_2a_1a_0$ in $GF(2^8)$, secondo la conversione vista precedentemente, e successivamente si calcola la stringa binaria $b_7b_6b_5b_4b_3b_2b_1b_0$ attraverso una trasformazione affine definita da

$$b_i = (a_i + a_{i+4} + a_{i+5} + a_{i+6} + a_{i+7} + c_i) \quad 0 \leq i \leq 7$$

dove $c_7c_6c_5c_4c_3c_2c_1c_0 = 01100011$.

[1] In seguito, per rappresentare il contenuto di un byte spesso, per semplicità, si utilizzerà la notazione esadecimale.

La rappresentazione matriciale della trasformazione affine è la seguente

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \ mod\ 2 \qquad (*)$$

Più precisamente, l'algoritmo **SubBytes** opera come segue:

Algoritmo 4.2. **(SubBytes** $(a_7a_6a_5a_4a_3a_2a_1a_0)$**)**

external FieldInv, BinaryToField, FieldToBinary
$z \leftarrow$ BinaryToField$(a_7a_6a_5a_4a_3a_2a_1a_0)$
if $z \neq 0$
    $(a_7a_6a_5a_4a_3a_2a_1a_0) \leftarrow$ FieldToBinary(z)
then $z \leftarrow$ FieldInv(z)
    $(c_7c_6c_5c_4c_3c_2c_1c_0) \leftarrow (01100011)$
for $i \leftarrow 0$ **to** 7
do $b_i \leftarrow (a_i + a_{i+4} + a_{i+5} + a_{i+6} + a_{i+7} + c_i)\ mod\ 2$
return $(b_7b_6b_5b_4b_3b_2b_1b_0)$

L'algoritmo esterno **FieldInv** è quello che permette di calcolare l'inverso dell'elemento in $GF(2^8)$ corrispondente alla stringa $a_7a_6a_5a_4a_3a_2a_1a_0$.

Un modo più immediato per determinare la stringa

$$b_7b_6b_5b_4b_3b_2b_1b_0$$

a partire dalla stringa

$$\alpha_7\alpha_6\alpha_5\alpha_4\alpha_3\alpha_2\alpha_1\alpha_0$$

è l'utilizzo della **Tabella 7** che racchiude le operazioni di calcolo dell'inverso in $GF(2^8)$ e della trasformazione affine $(*)$.

		Y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
X	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	BE	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tabella 7: AES S-box.

Supponiamo di voler cifrare il testo in chiaro x (convertito nel sistema esadecimale):

B95EF2E0E302E88C1D61A0B8AC82B65C.

Prima di applicare la **SubBytes** nella prima iterazione, al testo in chiaro viene inizialmente eseguita una somma XOR con la chiave K data in input, che supponiamo essere

274FE51AB701794D3ADF95B90F6C8BEE,

generando così la stringa

9E1117FA540391C127BE3501A3EE3DB2.

Terminata questa prima operazione, si esegue ora la sostituzione.

La **Tabella 7** viene applicata per ogni coppia XY di cifre esadecimali. L'immagine tramite l'S-box di XY è l'elemento della tabella posizionato nella riga X e colonna Y. Pertanto, nel caso della prima coppia di cifre di x partendo da sinistra, cioè di 9E, la tabella restituisce l'elemento 0B.

Procedendo allo stesso modo con le altre coppie del testo in chiaro, si genera così la stringa:

0B82F02D207B8178CCAE967C0A282737.

Vediamo in dettaglio come opera l'S-box.

Disponendo il testo in chiaro lungo le colonne di una matrice 4×4, il blocco corrispondente al messaggio **x** è

$9E$	54	27	$A3$
11	03	BE	EE
17	91	35	$3D$
FA	$C1$	01	$B2$

(4.1)

L'equivalente in binario è

(1001)(1110)	(0101)(0100)	(0010)(0111)	(1010)(0011)
(0001)(0001)	(0000)(0011)	(1011)(1110)	(1110)(1110)
(0001)(0111)	(1001)(0001)	(0011)(0101)	(0011)(1101)
(1111)(1010)	(1100)(0001)	(0000)(0001)	(1011)(0010)

Utilizzando la **Tabella 7**, SubBytes(x) è

$0B$	20	CC	$0A$
82	$7B$	AE	28
$F0$	81	96	27
$2D$	78	$7C$	37

.

Convertiamo ora ogni elemento della **Matrice** (4.1) in un elemento del campo $GF(2^8)$:

BinaryToField

$x^7+x^4+x^3+x^2+x$	$x^6+x^4+x^2$	x^5+x^2+x+1	x^7+x^5+x+1
x^4+1	$x+1$	$x^7+x^5+x^4+x^3+x^2+x$	$x^7+x^6+x^5+x^3+x^2+x$
x^4+x^2+x+1	x^7+x^4+1	$x^5+x^4+x^2+1$	$x^5+x^4+x^3+x^2+1$
$x^7+x^6+x^5+x^4+x^3+x$	x^7+x^6+1	1	$x^7+x^5+x^4+x$

L'inverso di ogni elemento della matrice precedente in $GF(2^8)$ è stato tabulato in

FieldInv

x^7+x^3+1	$x^6+x^3+x^2$	$x^7+x^6+x^3+1$	x^7+x^6+x+1
$x^7+x^5+x^4+x^2$	$x^7+x^6+x^5+x^4+x^2+x$	x^7+x^2+x	$x^4+x^3+x^2+x$
$x^6+x^4+x^3+x^2+x+1$	x^6+x^4+x	$x^5+x^4+x^3+1$	$x^7+x^5+x^4+x^3+x+1$
$x^6+x^5+x^4+x^3+x^2+1$	x^5+x^3	1	$x^4+x^3+x^2+x+1$

Il corrispondente in binario è

FieldToBinary

(1000)(1001)	(0100)(1100)	(1100)(1001)	(1100)(0011)
(1011)(0100)	(1111)(0110)	(1000)(0110)	(0001)(1110)
(0101)(1111)	(0110)(1010)	(0011)(1001)	(1011)(1011)
(0111)(1101)	(0010)(1000)	(0000)(0001)	(0001)(1111)

Applichiamo ora la trasformazione affine $(*)$ al primo elemento $9E$ di (4.1). Si determina così la stringa

$$(b_7b_6b_5b_4b_3b_2b_1b_0) = 00001011,$$

che in esadecimale corrisponde a 0B.

Applicando lo stesso algoritmo su tutti gli altri elementi di (4.1), la matrice in uscita del **SubBytes** è

00001011	00100000	11001100	00001010
10000010	01111011	10101110	00101000
11110000	10000001	10010110	00100111
00101101	01111000	01111100	00110111

ovvero, in esadecimale

$0B$	20	CC	$0A$
82	$7B$	AE	28
$F0$	81	96	27
$2D$	78	$7C$	37

(4.2)

4.4.2 ShiftRows

ShiftRows agisce sulle righe della matrice stato, pertanto l'i-esima riga, $i = 0, 1, 2, 3$, viene shiftata verso sinistra i volte.

Quindi la **ShiftRows** di

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$\mathbf{s_{0,3}}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$\mathbf{s_{1,3}}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$\mathbf{s_{2,3}}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$\mathbf{s_{3,3}}$

(4.3)

è la matrice

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$\mathbf{s_{0,3}}$
$s_{1,1}$	$s_{1,2}$	$\mathbf{s_{1,3}}$	$s_{1,0}$
$s_{2,2}$	$\mathbf{s_{2,3}}$	$s_{2,0}$	$s_{2,1}$
$\mathbf{s_{3,3}}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

(4.4)

In particolare, applicando l'operazione **ShiftRows** a (4.2), si ottiene

$0B$	20	CC	**0A**
$7B$	AE	**28**	82
96	**27**	$F0$	81
37	$2D$	78	$7C$

(4.5)

4.4.3 MixColumn

Per meglio comprendere l'operazione di **MixColumn**, è conveniente rappresentare le colonne della matrice in (4.5) come elementi dell'anello quoziente $\frac{GF(2^8)[y]}{\langle y^4+1\rangle}$.

Poichè ogni byte individua un unico elemento di $GF(2^8)$, allora ad ogni colonna della **Matrice** (4.4) può essere associato un polinomio di grado al più 3 a coefficienti in $GF(2^8)$. Siffatto polinomio individua un unico elemento dell'anello quoziente $\frac{GF(2^8)[y]}{\langle y^4+1\rangle}$ nella maniera usuale.

Quindi, per esempio, alla prima colonna nella **Matrice** (4.4) verrà associato l'elemento di $\frac{GF(2^8)[y]}{\langle y^4+1\rangle}$ individuato da

$$s_0(y) = \overline{s_{0,0}}\, y^3 + \overline{s_{1,1}}\, y^2 + \overline{s_{2,2}}\, y + \overline{s_{3,3}},$$

dove $\overline{s_{i,i}}$ è il corrispondente in $GF(2^8)$ di $s_{i,i}$.

I polinomi corrispondenti alle colonne della matrice in (4.5) (o anche (4.4)) sono moltiplicati per il polinomio

$$d(y) = \overline{03}y^3 + \overline{01}y^2 + \overline{01}y + \overline{02}.$$

Pertanto, l'output dell'operazione **MixColumn** è

$$s'(y) = d(y)s(y) \qquad mod\ (y^4 + 1).$$

Come nell'operazione **SubBytes**, la moltiplicazione può essere espressa in forma matriciale come segue

$$\begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} \overline{02} & \overline{03} & \overline{01} & \overline{01} \\ \overline{01} & \overline{02} & \overline{03} & \overline{01} \\ \overline{01} & \overline{01} & \overline{02} & \overline{03} \\ \overline{03} & \overline{01} & \overline{01} & \overline{02} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}. \qquad (**)$$

Quindi, il risultato di **MixColumn** applicato a (4.4) è dato da

$$\begin{array}{|c|c|c|c|} \hline \overline{02} & \overline{03} & \overline{01} & \overline{01} \\ \hline \overline{01} & \overline{02} & \overline{03} & \overline{01} \\ \hline \overline{01} & \overline{01} & \overline{02} & \overline{03} \\ \hline \overline{03} & \overline{01} & \overline{01} & \overline{02} \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|} \hline \overline{s_{0,0}} & \overline{s_{0,1}} & \overline{s_{0,2}} & \overline{s_{0,3}} \\ \hline \overline{s_{1,1}} & \overline{s_{1,2}} & \overline{s_{1,3}} & \overline{s_{1,0}} \\ \hline \overline{s_{2,2}} & \overline{s_{2,3}} & \overline{s_{2,0}} & \overline{s_{2,1}} \\ \hline \overline{s_{3,3}} & \overline{s_{3,0}} & \overline{s_{3,1}} & \overline{s_{3,2}} \\ \hline \end{array} \qquad (4.6)$$

Lo pseudo-codice di **MixColumn** è il seguente:

Algoritmo 4.3. (**MixColumn**(c))
external FieldMult, BinaryToField, FieldToBinary
for $i \leftarrow$ **to** 3
do $t_i \leftarrow$ BinaryToField($s_{i,c}$)
$u_0 \leftarrow \text{FieldMult}(x, t_0) \oplus \text{FieldMult}(x+1, t_1) \oplus t_2 \oplus t_3$
$u_1 \leftarrow \text{FieldMult}(x, t_1) \oplus \text{FieldMult}(x+1, t_2) \oplus t_3 \oplus t_0$
$u_2 \leftarrow \text{FieldMult}(x, t_2) \oplus \text{FieldMult}(x+1, t_3) \oplus t_0 \oplus t_1$
$u_3 \leftarrow \text{FieldMult}(x, t_3) \oplus \text{FieldMult}(x+1, t_0) \oplus t_1 \oplus t_2$
for $i \leftarrow 0$ **to** 3
do $s_{i,c} \leftarrow \text{FieldToBinary}(u_i)$

Osservando gli input, l'algoritmo **FieldMult** moltiplica una costante con una variabile. Infatti, il primo argomento è il polinomio x o $x+1$ che corrisponde all'elemento $\overline{02}$ e $\overline{03}$ del campo $GF(2^8)$, rispettivamente.

In binario, la moltiplicazione di t_i per x è equivalente ad uno shift verso sinistra di t_i, mentre la moltiplicazione per $x+1$ corrisponde ad uno shift verso sinistra di t_i a cui viene successivamente sommato t_i. Inoltre, l'operazione di XORING corrisponde all'usuale somma modulo 2 componente per componente.

Applicare l'operazione **MixColumn** al risultato di **ShiftRows**

$\overline{0B}$	$\overline{20}$	$\overline{CC}$	$\overline{0A}$
$\overline{7B}$	$\overline{AE}$	$\overline{28}$	$\overline{82}$
$\overline{96}$	$\overline{27}$	$\overline{F0}$	$\overline{81}$
$\overline{37}$	$\overline{2D}$	$\overline{78}$	$\overline{7C}$

(4.7)

corrisponde a trasformare ogni colonna di (4.7) mediante (∗∗) ottenendo in questo modo

$\overline{3A}$	$\overline{A3}$	$\overline{73}$	$\overline{74}$
$\overline{6B}$	$\overline{23}$	$\overline{EF}$	$\overline{F1}$
$\overline{1E}$	$\overline{B7}$	$\overline{97}$	$\overline{15}$
$\overline{9E}$	$\overline{B3}$	$\overline{67}$	$\overline{E5}$

(4.8)

L'equivalente in binario di (4.8) è

00111010	10100011	01110011	01110100
01101011	00100011	11101111	11110001
00011110	10110111	10010111	00010101
10011110	10110011	01100111	11100101

(4.9)

.

Siccome le operazioni sono eseguite in $GF(2^8)$, spieghiamo in dettaglio come da (4.7) si ottiene (4.8). Lo faremo solo per la prima colonna siccome per le altre si procede in modo analogo. Vale che

t_0	$\overline{0B}$	x^3+x+1
t_1	$\overline{7B}$	$x^6+x^5+x^4+x^3+x+1$
t_2	$\overline{96}$	$x^7+x^4+x^2+x$
t_3	$\overline{37}$	$x^5+x^4+x^2+x+1$

Tenendo presente la riduzione modulo $p(x) = x^8+x^4+x^3+x+1$ si ha

$$\begin{array}{l}
\overline{u_{0,0}} = x(x^3+x+1)+(x+1)(x^6+x^5+x^4+x^3+x+1)+(x^7+x^4+x^2+x)+(x^5+x^4+x^2+x+1) \\
\overline{u_{1,0}} = x(x^6+x^5+x^4+x^3+x+1)+(x+1)(x^7+x^4+x^2+x)+(x^5+x^4+x^2+x+1)+(x^3+x+1) \\
\overline{u_{2,0}} = x(x^7+x^4+x^2+x)+(x+1)(x^5+x^4+x^2+x+1)+(x^3+x+1)+(x^6+x^5+x^4+x^3+x+1) \\
\overline{u_{3,0}} = x(x^5+x^4+x^2+x+1)+(x+1)(x^3+x+1)+(x^6+x^5+x^4+x^3+x+1)+(x^7+x^4+x^2+x)
\end{array}$$

Pertanto,

$\overline{u_{0,0}}$	$x^5+x^4+x^3+x$	$\overline{3A}$	00111010
$\overline{u_{1,0}}$	$x^6+x^5+x^3+x+1$	$\overline{6B}$	01101011
$\overline{u_{2,0}}$	$x^4+x^3+x^2+x$	$\overline{1E}$	00011110
$\overline{u_{3,0}}$	$x^7+x^4+x^3+x^2+x$	$\overline{9E}$	10011110

corrisponde alla prima colonna di (4.8) e di (4.9).

4.4.4 AddRoundKey

Quest'ultima operazione consiste in un'operazione XOR tra lo stato corrente calcolato w^i e la chiave K^i. Tale operazione bit-a-bit corrisponde all'addizione in $GF(2)$.

4.4.5 KeyExpansion

Prima di iniziare il processo di cifratura, il crittosistema prevede la creazione di una lista di N_r+1 sottochiavi $(K^0, K^1, \ldots, K^{N_r})$ a partire da una chiave iniziale segreta K data in input attraverso l'algoritmo spiegato qui di seguito (una per ogni iterazione più quella necessaria per lo sbiancamento nel primo **AddRoundKey**). La concatenazione delle sottochiavi è chiamata **chiave espansa**. Ogni sottochiave è una quadrupla di **word** (ciascuna costituita da 4 byte):

$$(w[4i], w[4i+1], w[4i+2], w[4i+3]).$$

Al suo interno vengono eseguite due operazioni:

1. **RotWord:** opera uno shift a sinistra di un byte di una word.
2. **SubWord:** applica l'S-box definito per l'AES su ognuno dei 4 byte di ogni word data in input.

L'input di **KeyExpansion** è una chiave K di lunghezza fissata che si presenta nella forma di una stringa $K_0, \ldots, K_j$ byte con $j = 15$ o 23 o 31 a seconda che K sia rispettivamente di 128 o 192 o 256 bit. L'algoritmo poi restituisce una matrice di elementi $w[i]$. Le due operazioni lavorano su ogni singolo elemento $w[i]$, pertanto richiedono come input una word. Se $B_0B_1B_2B_3$ è una word, con B_i un byte, allora

- **RotWord**$(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0)$
- **SubWord**$(B_0, B_1, B_2, B_3) = (B'_0, B'_1, B'_2, B'_3)$, dove B'_i si ottiene da B_i applicando l'inverso nel campo $GF(2^8)$ di B_i e poi la trasformazione affine definita in $(*)$.

Lo pseudocodice dell'algoritmo **KeyExpansion** è descritto di seguito:

Algoritmo 4.4. **(KeyExpansion**(K)**)**

external RotWord, SubWord
$RC[1] \leftarrow 01000000$
$RC[2] \leftarrow 02000000$
$RC[3] \leftarrow 04000000$
$RC[4] \leftarrow 08000000$
$RC[5] \leftarrow 10000000$
$RC[6] \leftarrow 20000000$
$RC[7] \leftarrow 40000000$
$RC[8] \leftarrow 80000000$
$RC[9] \leftarrow 1B000000$
$RC[10] \leftarrow 36000000$
for $i \leftarrow 0$ **to** 3
do $w[i] \leftarrow (chiave[4i], chiave[4i+1], chiave[4i+2], chiave[4i+3])$
for $i \leftarrow 4$ **to** 43
do
$temp \leftarrow w[i-1]$
if $i \equiv \mathbf{0} \ mod \ 4$
then $temp \leftarrow \text{SubWord}(\text{RotWord}(temp)) \oplus RC[i/4]$
else $w[i] \leftarrow w[i-4] \oplus temp$
return $(w[0], \ldots, w[43])$

L'algoritmo inizializza dieci word costanti denominate $RC[i]$ per $i = 1, \ldots, 10$. Successivamente, suddivide la chiave data in input in 4 word. Quindi,

$$K = (w[0], w[1], w[2], w[3]).$$

Le word $w[i]$ successive o vengono determinate da

$$w[i-4] \oplus \textbf{SubWord}(\textbf{RotWord}(w[i-1])) \oplus RC[i/4],$$

o vengono determinate attraverso $w[i-4] \oplus w[i-1]$, a seconda che i sia un multiplo di 4 o meno, rispettivamente.

La chiave espansa i-esima è

$$K^i = (w[4i], w[4i+1], w[4i+2], w[4i+3]),$$

dove $i = 0, \ldots, N_r$. In particolare, $K^0 = K$. Sia

$$g(w) = \text{SubWord}(\text{RotWord}(w)) \oplus RC[i/4],$$

per i multiplo di 4, allora la struttura dell'algoritmo **KeyExpansion** è descritta nella seguente figura.

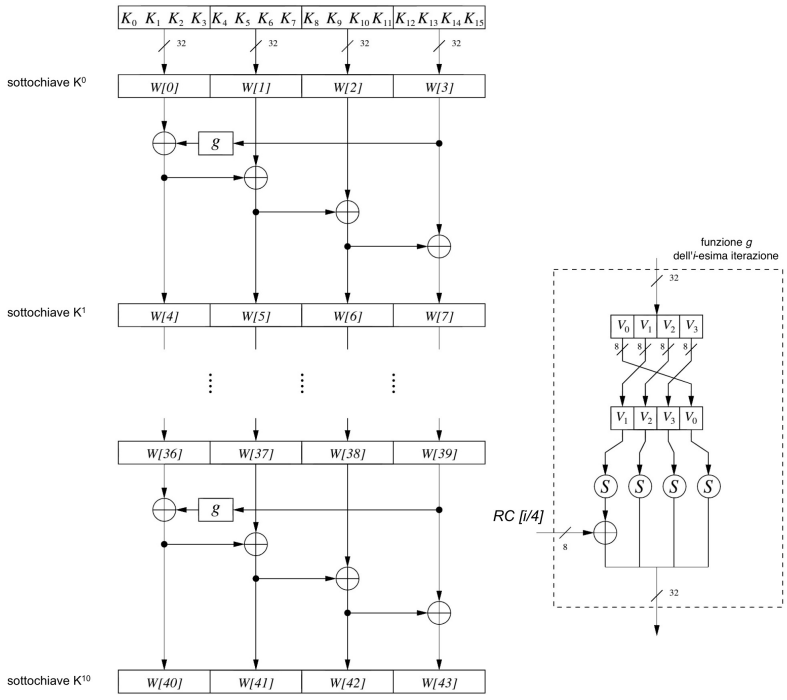


Figura 4.4: Struttura dell'operazione KeyExpansion

Forniamo un esempio concreto di come la **KeyExpansion** opera su una chiave K di 128 bit. Per semplicità useremo la notazione esadecimale ricordando che ogni byte corrisponde ad una coppia di cifre esadecimali.

Sia $K = (K_0, K_1, \ldots, K_{15})$ dove

K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
27	4F	E5	1A	B7	01	79	4D	3A	DF	95	B9	0F	6C	8B	EE

L'algoritmo genera 11 sottochiavi immagazzinate nella matrice che rappresenta la chiave espansa. Analizziamo nel dettaglio solo la creazione della seconda sottochiave K^1, il procedimento è analogo per le altre sottochiavi.

La sottochiave $K^0 = K$ è

$$(w[0], w[1], w[2], w[3]) = (274FE51A,\ B701794D,\ 3ADF95B9,\ 0F6C8BEE).$$

Dalla definizione di **RotWord** e **SubWord** segue che

$$\text{RotWord}(w[3]) = 6C8BEE0F$$
$$\text{SubWord}(\text{RotWord}(w[3])) = 503D2876.$$

Quindi

$$w[4] = (503D2876 \oplus 01000000) \oplus w[0] = 7672CD6C$$
$$w[5] = w[1] \oplus w[4] = C173B421$$
$$w[6] = w[2] \oplus w[5] = FBAC2198$$
$$w[7] = w[3] \oplus w[6] = F4C0AA76,$$

e pertanto K^1 è

$$(w[4], w[5], w[6], w[7]) = (7672CD6C,C173B421,FBAC2198,F4C0AA76).$$

Quindi,

$K^0 =$

w[0]	w[1]	w[2]	w[3]
27	B7	3A	0F
4F	01	DF	6C
E5	79	95	8B
1A	4D	B9	EE

$K^1 =$

w[4]	w[5]	w[6]	w[7]
76	C1	FB	F4
72	73	AC	C0
CD	B4	21	AA
6C	21	98	76

Procedendo in modo analogo per $K^2, \ldots, K^{10}$ si ottiene la tabella:

K^2 =

w[8]	w[9]	w[10]	w[11]
CE	0F	F4	00
DE	AD	01	C1
F5	41	60	CA
D3	F2	6A	1C

K^3 =

w[12]	w[13]	w[14]	w[15]
B2	BD	49	49
AA	07	06	C7
69	28	48	82
B0	42	28	34

K^4 =

w[16]	w[17]	w[18]	w[19]
7C	C1	88	C1
B9	BE	B8	7F
71	59	11	93
8B	C9	E1	D5

K^5 =

w[20]	w[21]	w[22]	w[23]
BE	7F	F7	36
65	DB	63	1C
72	2B	3A	A9
F3	3A	DB	0E

K^6 =

w[24]	w[25]	w[26]	w[27]
02	7D	8A	BC
B6	6D	0E	12
D9	F2	C8	61
F6	CC	17	19

K^7 =

w[28]	w[29]	w[30]	w[31]
8B	F6	7C	C0
59	34	3A	28
0D	FF	37	56
93	5F	48	51

K^8 =

w[32]	w[33]	w[34]	w[35]
3F	C9	B5	75
E8	DC	E6	CE
DC	23	14	42
29	76	3E	6F

K^9 =

w[36]	w[37]	w[38]	w[39]
AF	66	D3	A6
C4	18	FE	30
74	57	43	01
B4	C2	FC	93

K^{10} =

w[40]	w[41]	w[42]	w[43]
9D	FB	28	8E
B8	A0	5E	6E
A8	FF	BC	BD
90	52	AE	3D

Ciò completa la descrizione del cifrario AES.

4.5 Esempio di cifratura

Presentiamo ora un esempio di cifratura integrale in cui la lunghezza della chiave K è pari a 128 bit ed il numero delle iterazioni è 10.
I dati in input sono:

Blocco iniziale = B9 5E F2 E0 E3 02 E8 8C 1D 61 A0 B8 AC 82 B6 5C

Chiave K = 27 4F E5 1A B7 01 79 4D 3A DF 95 B9 0F 6C 8B EE

Iter.	Stato	SubBytes	ShiftRows	MixColumns		AddRoundKey
0	B9 E3 1D AC 5E 02 61 82 F2 E8 A0 B6 E0 8C B8 5C				⊕	27 B7 3A 0F 4F 01 DF 6C E5 79 95 8B 1A 4D B9 EE
1	9E 54 27 A3 11 03 BE EE 17 91 35 3D FA C1 01 B2	0B 20 CC 0A 82 7B AE 28 F0 81 96 27 2D 78 7C 37	0B 20 CC 0A 7B AE 28 82 96 27 F0 81 37 2D 78 7C	3A A3 73 74 6B 23 EF F1 1E B7 97 15 9E B3 67 E5	⊕	76 C1 FB F4 72 73 AC C0 CD B4 21 AA 6C 21 98 76

Pertanto, applicando l'intero Crittosistema **AES** al messaggio in chiaro

B95EF2E0E302E88C1D61A0B8AC82B65C,

si determina il seguente messaggio cifrato

6E717F1C64C34C4F95135C4C74CEE503.

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.

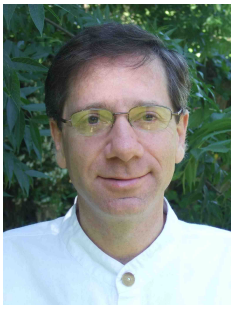



Figura 5.2: Mihir Bellare (1962) e Phillip Rogaway (1962)

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?"

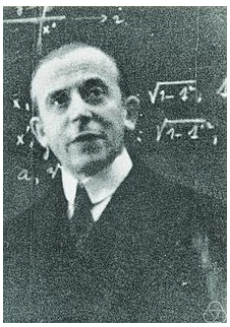


Figura 5.3: Richard von Mises (1883 – 1953)

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**.

□

6 Complessità computazionale di un algoritmo

6.1 Rappresentazione di un intero in base b

Definizione 6.1. **(Rappresentazione di n in base b)**
Siano n, b interi non negativi, allora $n = d_{k-1}b^{k-1} + ... + d_1 b + d_0$, dove $0 \leq d_{k-1}, ...d_1, d_0 \leq b-1$. La notazione $(d_{k-1}...d_1 d_0)_b$ indica la **rappresentazione di n in base b** e le gli interi $d_{k-1}, ...d_1, d_0$ sono le **cifre di n in base b**.

Se $b = 2$, allora le cifre in binario si dicono **bit** (binary digit).

Esempio 6.2. Esprimiamo $n = 1000$ in base 2, 7 e 26.

1. $b = 2$, allora $n = 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3$ e quindi $n = 1111101000$.
2. $b = 7$, allora $n = 2 * 7^3 + 6 * 7^2 + 2 * 7 + 6$ e $n = (2626)_7$.
3. $b = 26$, usando le lettere dell'alfabeto inglese come cifre $A = 0, \cdots, Z = 25$, si ha $n = (BLC)_{26}$.

Si noti che se $n = (d_{k-1}...d_1 d_0)_b$, vale che $b^{k-1} \leq n < b^k$ ed il numero delle cifre di n in base b è

$$k = \lfloor \log_b n \rfloor + 1.$$

Esempio 6.3. Dall'esempio precedente segue che $n = 1000$ in base 2, 7 e 26 ha il seguente numero di cifre:

1. $n = 1111101000$ e $\lfloor \log_2 10^3 \rfloor + 1 = 10$.
2. $n = (2626)_7$ e $\lfloor \log_7 10^3 \rfloor + 1 = 4$.
3. $n = (BLC)_{26}$ e $\lfloor \log_{26} 10^3 \rfloor + 1 = 3$.

6.2 Complessità temporale di un algoritmo

L'efficienza di un algoritmo nella risoluzione di un problema si misura in base a

1. Il **tempo** richiesto per eseguire le operazioni elementari.
2. Lo **spazio** necessario per memorizzare e manipolare dati.

Il tempo è la risorsa più importante deve essere indipendente dalle performance del calcolatore utilizzato. Un modo oggettivo per la misura del tempo di calcolo di un algoritmo è quello di esprimerlo in funzione della dimensione n dei dati in input.

Definizione 6.4. **(Complessità computazionale di un algoritmo)** La **complessità computazionale (temporale)** di un algoritmo è una funzione $T(n)$ dove n è la dimensione dei dati di ingesso.

Siccome vale che $\lim_{n\to+\infty} T(n) = +\infty$, per stimare T è importante l' ordine di infinito di T. Per fare ciò si consideri la seguente definizione.

Definizione 6.5. **(Notazione O-grande)** Siano $f, g : \mathbb{N}^r \longrightarrow \mathbb{R}$, allora $f = O(g)$ se, e solo se, esistono due costanti $B, C \in \mathbb{R}$ tali che per ogni $j = 1, ...r$, se $n_j > B$ allora

$$f(n_1, ..., n_r) < Cg(n_1, ..., n_r).$$

Esempio 6.6. Valgono i seguenti fatti:

1. $(r = 1)$, se $f(n) = 2n^2 + 3n - 3$, allora $f(n) < 3n^2$ e quindi $f = O(n^2)$.
2. Siano $f, g : \mathbb{N} \longrightarrow \mathbb{R}$, se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \ell$ e $\ell \in \mathbb{R}$, allora $f = O(g)$. Infatti, fissato $\varepsilon = \ell/2$ per esempio, esiste $B > 0$ tale che per ogni $n > B$ risulta $\left|\frac{f(n)}{g(n)} - \ell\right| < \frac{\ell}{2}$ e quindi $f(n) < \frac{3}{2}\ell g(n)$.
3. Per ogni $\varepsilon > 0$ vale che $\ln n = O(n^\varepsilon)$. Segue da (1) tenendo presente che $\lim_{n\to+\infty} \frac{\ln n}{n^\epsilon} = 0$, per esempio, per la Teorema di de l'Hôpital.
4. Sia $f(n, b) = \lfloor \log_b n \rfloor + 1$, allora $f(n, b) < 2 \log_b n$ e quindi $f(b, n) = O(\log_b n)$.

Definizione 6.7. **(Complessità computazionale polinomiale di un algoritmo)**
Un algoritmo che ha come input gli interi $n_1, n_2..., n_r$ di $k_1, k_2..., k_r$ bit, rispettivamente, si dice avere **complessità computazionale polinomiale**, se esistono degli interi $d_1, d_2..., d_r$ tali che

$$T(n) = O(k_1^{d_1} k_2^{d_2} \cdots k_r^{d_r}).$$

6.3 Bit Operazioni

In questo sezione intendiamo stimare il tempo di esecuzione di semplici algoritmi utilizzati nell'ambito della teoria dei numeri.

6.3.1 Somma di due numeri di lunghezza binaria k.

Per esempio, si opera come segue.

In realtà dobbiamo ripetere questo procedimento k volte:

1. Se entrambi i bit sono 0 e non ci sta riporto, si segna 0 e si prosegue.
2. Se entrambi i bit sono 0 e ci sta riporto di 1, oppure uno dei due bit è 0, altro è 1 e non ci sta riporto, allora si scrive 1 e si prosegue
3. Se uno dei due bit è 0, altro è 1 e ci sta il riporto di 1, oppure entrambi sono 1 e non ci sta riporto, allora si scrive 0 e si mette il riporto di 1 alla colonna successiva.
4. Se entrambi i bit sono 1 e ci sta il riporto di 1, si scrive 1 e mette il riporto di 1 alla colonna successiva.

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ } \\ 1111000 \\ +\ \underline{0011110} \\ 10010110 \end{array}$$

Ognuno di questi quattro passaggi è chiamato **operazione bit**. Quindi, sommare due numeri, entrambi di lunghezza binaria k, richiede k operazioni.

6.3.2 Prodotto di due numeri n e m di lunghezza binaria k e ℓ, rispettivamente, con $k \geq \ell$.

Molte delle stime per il calcolo della complessità computazionale di operazioni elementari fatte qui di seguito non sono ottimali, ma hanno tuttavia un valore didascalico. Per esempio, si moltiplichi 11101 per 1101.

$$\begin{array}{r} 11101 \\ \underline{\quad 1101} \\ 11101 \\ 111010 \\ \underline{11101\quad\ } \\ 101111001 \end{array}$$

Dalla figura si evince che, se $\ell' \leq \ell$ è il numero delle cifre non nulle di m, allora si riportano ℓ' copie di n, ognuna shiftata verso sinistra di una posizione rispetto alla precedente.

Siccome vogliamo valutare la moltiplicazione di n per m, non possiamo sommare le ℓ' copie shifate di n ma dobbiamo procedere in modo induttivo: sommiamo le prime due, poi sommiamo ogni riga alla somma parziale delle precedenti. Quindi eseguiamo $\ell' - 1$ addizioni.

Siccome ognuna di queste prevede k operazioni bit, allora

$$T(n \times m) < k\ell < k^2.$$

Esercizio 6.8. Provare che $T(\lfloor \sqrt{n} \rfloor) = O(\log_2 n)^3$.

Dimostrazione. Supponiamo che n abbia $k+1$ bit. Se $m = \lfloor \sqrt{n} \rfloor$, allora $m = 2^{\lfloor k/2 \rfloor} + \sum_{i=0}^{\lfloor k/2 \rfloor - 1} x_i 2^i$.
Allora si opera come segue:

1. Per ogni $j = 1, ..., \lfloor k/2 \rfloor$ si testa $m^2 \leq n$ per m ottenuto in corrispondenza delle $(\lfloor k/2 \rfloor + 1)$-upla $\left(x_{\lfloor k/2 \rfloor}, x_{\lfloor k/2 \rfloor - 1}, ..., x_0\right)$ tale che $x_{\lfloor k/2 \rfloor - e} = 1$ per $e \leq j$ e $x_{\lfloor k/2 \rfloor - e} = 0$ per $e \geq j + 1$.
2. Quindi si eseguono al più $\lfloor k/2 \rfloor + 1$ moltiplicazioni di numero con sè stesso, avente al più $\lfloor k/2 \rfloor$ cifre.
3. Il numero delle bit operazioni eseguite è al più $(\lfloor k/2 \rfloor + 1)(\lfloor k/2 \rfloor)^2 \leq k^3$.

Pertanto $T(\lfloor \sqrt{n} \rfloor) = O(\log_2 n)^3$.

□

Esercizio 6.9. Provare che $T(n!) = O((n-2)n\left(\lfloor \log_2 n \rfloor + 1\right)^2)$.

Dimostrazione. Per calcolare $n! = 2 \cdot 3 \cdots (n-1) \cdot n$ si procede come segue. Si calcola $2 \cdot 3$, il risultato lo si moltiplica per 4, il risultato per 5. Essendo i numeri $n-1$, allora il numero delle moltiplicazioni che si esegue è $n-2$.

1. Per $j = 1, \ldots, n-1$, al passo $(j-1)$-esimo, si moltiplica $j!$ per $j+1$. Se $k = \lfloor \log_2 n \rfloor + 1$, per ogni $2 \leq j \leq n$ risulta $\lfloor \log_2 j \rfloor + 1 \leq k$ e $\lfloor \log_2 (j!) \rfloor + 1 \leq nk$.

2. Il numero delle bit operazioni necessarie per calcolare $j!$ per $j+1$ è minore di $nk \cdot k = nk^2$.

3. Siccome si eseguono $n-2$ moltiplicazioni nel punto **(2)**, allora
$$T(n!) = O((n-2)nk^2).$$

4. Per $n \to +\infty$ vale che $(n-2)n\left(\lfloor \log_2 n \rfloor + 1\right)^2 \simeq n^2 \left(\log_2 n\right)^2$.

□

Esercizio 6.10. Provare che $T(\binom{n}{m}) = O(m^2 \left(\log_2 n\right)^2)$.

Dimostrazione. Siccome $\binom{n}{m} = \binom{n}{n-m}$ possiamo assumere $m \leq n/2$. Inoltre, poiché
$$\binom{n}{m} = \frac{D_{n,m}}{2 \cdot 3 \cdots (m-1) \cdot m},$$
dove $D_{n,m} = n \cdot (n-1) \cdots (n-m+1)$, per determinare $\binom{n}{m}$ è necessario eseguire $m-1$ moltiplicazioni e successivamente $m-1$ divisioni.

1. Per $j = 1, \ldots, m-1$, al passo $(j-1)$-esimo si moltiplica $D_{n,j}$ per $n-j$, dove $D_{n,j} \leq D_{n,m} \leq n^m$.

2. Se $k = \lfloor \log_2 n \rfloor + 1$, allora
$$\lfloor \log_2 D_{n,j} \rfloor + 1 \leq \lfloor \log_2 D_{n,m} \rfloor + 1 \leq \lfloor \log_2 n^m \rfloor + 1 = mk.$$

3. Per $j = 1, \ldots, m-1$, al passo $(j-1)$-esimo si eseguono al più mk^2 bit operazioni. Pertanto il numero delle bit operazioni per determinare $D_{n,m}$ è al più $(m-1)mk^2$.

4. Determinato $D_{n,m}$, si eseguono $m-1$ divisioni tra $D_{n,m}$ e i $2 \leq e \leq m \leq n/2$, ognuna delle quali impiega al più $(mk)\,k = mk^2$ bit operazioni.

   Quindi il numero delle bit operazioni necessario per dividere $D_{n,m}$ con $m!$ è $(m-1)mk^2$.

5. $T(\binom{n}{m}) < (m-1)mk^2 + (m-1)mk^2 < 2m^2k^2$ e $2m^2k^2 \simeq 2m^2 \left(\log_2 n\right)^2$ per $n \to +\infty$, che è l'asserto.

□

Esercizio 6.11. Provare che $T(n_b) = O((\ln n)^2)$.

Dimostrazione. Per convertire un intero scritto in binario n e avente k bit in base b bisogna dividere per $b = (b_{\ell-1}...b_0)_2$. Il resto d_0 sarà uno degli interi da 0 a $b-1$. Quindi $n = n_1 b + d_0$. Ora rimpiazziamo n con n_1 e ripetiamo l'operazione.

1. Il numero delle divisioni da eseguire è uguale al numero delle cifre in base b di n che è
$$\left\lfloor \frac{\ln n}{\ln b} \right\rfloor + 1 = \left\lfloor \frac{\ln n}{\ln 2} \cdot \left(\frac{\ln b}{\ln 2} \right)^{-1} \right\rfloor + 1 = O(k/\ell).$$
2. Il numero dei bit operazioni impiegati per eseguire la divisione di numeri (i quozienti) minori o uguali a n con b è $O(k\ell)$.
3. $T(n_b) = O(k/\ell) \cdot O(k\ell) = O(k^2) = O((\ln n)^2)$.

□

Si noti che il tempo non dipende dalla grandezza della base b scelta. Ci ò avviene perché il maggior tempo utilizzato per determinare ogni cifra in base b è compensato dal minor numero di cifre da determinare.

Quindi il numero delle bit operazioni impiegate per eseguire la divisione è k^2. Il numero dei passai da eseguire è uguale al numero delle cifre decimale di n che è $\left\lfloor \frac{\ln n}{\ln 10} \right\rfloor + 1 = O(k)$, essendo $k = \left\lfloor \frac{\ln n}{\ln 2} \right\rfloor + 1$ avente k bit.

Esercizio 6.12. Siano $P(X) = \sum_{i=0}^{n_1} a_i X^i$ e $Q(X) = \sum_{j=0}^{n_2} b_j X^j$, $n_2 \le n_1$, tali che $a_i, b_j \in \mathbb{R}$ e $a_i, b_j \le m$, allora $T(P(X)Q(X)) = O(n^2 (\ln m)^2)$, dove n denota n_1.

Dimostrazione. Sia $R(X) = P(X)Q(X)$, allora $R(X) = \sum_{v=0}^{n_1+n_2} c_v X^v$, dove $c_v = \sum_{i+j=v} a_i b_j$. Il numero degli addendi in c_v è al più uguale al numero dei b_j che è $n_2 + 1$. Quindi per determinare c_v occorrono al più n_2 addizioni e $n_2 + 1$ moltiplicazioni.

1. Per eseguire le n_2 moltiplicazioni occorrono $(n_2 + 1)(\lfloor \log_2 m \rfloor + 1)^2$ bit operazioni.
2. Poiché $a_i b_j \le m^2$ allora $c_v \le n_2 m^2$ e quindi ogni somma parziale è minore o uguale di $n_2 m^2$. Quindi il numero delle bit operazioni necessario per eseguire una addizione è minore o uguale al numero delle bit operazioni necessario per eseguire $n_2 m^2 + m^2$, che è $(\lfloor \log_2 n_2 m^2 \rfloor + 1)$. Quindi il per eseguire le n_2 addizioni sono necessarie al più $n_2(\lfloor \log_2 n_2 m^2 \rfloor + 1)$.
3. Il numero totale delle bit operazioni per determinare c_v è al più
$$(n_2 + 1)(\lfloor \log_2 m \rfloor + 1)^2 + n_2(\lfloor \log_2 n_2 m^2 \rfloor + 1).$$
4. il numero dei c_v è al più $n_1 + n_2 + 1$, quindi il numero N di bit operazioni utilizzati per determinare $R(X)$ è
$$N \le (n_1 + n_2 + 1)\left[(n_2 + 1)(\lfloor \log_2 m \rfloor + 1)^2 + n_2(\lfloor \log_2 n_2 m^2 \rfloor + 1)\right].$$

5. Siccome $n_2 \leq n_1$, posto $n = n_1$, per $m \geq 16$ and $m \geq \sqrt{n_2}$ che si verificano nella realtà essendo $m \to +\infty$

$$N \leq (2n+1)\left[(n+1)(\log_2 m + 1)^2 + n(3\log_2 m + 1)\right]$$
$$\simeq 2n^2 (\log_2 m)^2 = \frac{2}{\ln 2} n^2 (\ln m)^2$$

per $n, m \to +\infty$.

Pertanto $T(P(X)Q(X)) = O(n^2 (\ln m)^2)$.

□

6.3.3 Algoritmo Euclideo

L'**Algoritmo Euclideo** è usato per il calcolo del massimo comune divisore tra due interi a e b.

Per semplicità assumiamo $0 < b \leq a$. Posto $r_0 = a$ e $r_1 = b$, esso opera come segue:

$$\begin{array}{ll} r_0 = q_1 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_2 r_2 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{m-2} = q_{m-1} r_{m-1} + r_m & 0 < r_m < r_{m-1} \\ r_{m-1} = q_m r_m & \end{array}$$

Lo pseudocodice dell'Algoritmo Euclideo è

Algoritmo 6.13. **(Algoritmo Euclideo (a, b))**

$r_0 \longleftarrow a$
$r_1 \longleftarrow b$
$m \longleftarrow 1$
while $r_m \neq 0$
do $\begin{cases} q_m \longleftarrow \left\lfloor \frac{r_{m-1}}{r_m} \right\rfloor \\ r_{m+1} \longleftarrow r_{m-1} - q_m r_m \\ m \longleftarrow m + 1 \end{cases}$
$m \longleftarrow m - 1$
return $(q_1, ..., q_m; r_m)$
comment $r_m = \gcd(a, b)$

Proposizione 6.14. Valgono i seguenti fatti:

(*i*) $\gcd(a,b) = r_m$.
(*ii*) $m \leq 2\lfloor \log_2 a \rfloor + 1$.
(*iii*) Se $a > b$, $T(\gcd(a,b)) = O(\ln^3 a)$.

Dimostrazione.

(*i*) Se t è un intero, poiché $r_i = q_{i+1}r_{i+1} + r_{i+2}$ allora risulta

$$t \mid \gcd(r_i, r_{i+1}) \iff t \mid r_i \text{ e } t \mid r_{i+1} \iff t \mid r_{i+1} \text{ e } t \mid r_{i+2}$$
$$\iff t \mid \gcd(r_{i+1}, r_{i+2}).$$

Pertanto

$$\gcd(a,b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m.$$

(*ii*) Se $r_{i+1} \leq \frac{1}{2}r_i$, allora $r_{i+2} < r_{i+1} \leq \frac{1}{2}r_i$. Se, invece $r_{i+1} > \frac{1}{2}r_i$, allora si ha $r_i = r_{i+1} + r_{i+2}$ (i.e. $q_{i+1} = 1$) e quindi, $r_{i+2} = r_i - r_{i+1} < r_i - \frac{1}{2}r_i = \frac{1}{2}r_i$.

Quindi, per ogni $i = 0, \ldots m-2$ vale che $r_{i+2} < \frac{1}{2}r_i$. Pertanto, $1 \leq r_m < \frac{1}{2^{\lfloor m/2 \rfloor}} r_0$ e quindi $\lfloor m/2 \rfloor < \log_2 a$. Ciò implica $m \leq 2\lfloor \log_2 a \rfloor + 1$

(*iii*) Poichè $r_i \leq a$, ogni divisione tra r_i e r_{i+1} impiega $O(\ln^2 a)$ bit operazioni, e poichè il numero delle divisioni da effettuare è $m \leq 2\lfloor \log_2 a \rfloor + 1 = O(\ln a)$, allora il tempo impiegato dall'**Algoritmo 6.13** per determinare $\gcd(a,b)$ è $O(\ln^2 a)O(\ln a) = O(\ln^3 a)$.

□

Remark 6.15. In realtà si prova che, se $a > b$, il tempo impiegato dall'**Algoritmo 6.13** per determinare $\gcd(a,b)$ è $O(\ln^2 a)$.

Siano a e b interi tali che $0 < b \leq a$, e siano $r_1, \ldots, r_m$ e $q_1, \ldots, q_m$ gli interi generati dall'algoritmo euclideo. Si definiscano gli interi $t_1, \ldots, t_m$ e $s_1, \ldots, s_m$ come segue:

$$t_j = \begin{cases} 0 & \text{se } j = 0 \\ 1 & \text{se } j = 1 \\ t_{j-2} - q_{j-1}t_{j-1} & \text{se } j \geq 2 \end{cases}$$

$$s_j = \begin{cases} 1 & \text{se } j = 0 \\ 0 & \text{se } j = 1 \\ s_{j-2} - q_{j-1}s_{j-1} & \text{se } j \geq 2 \end{cases}$$

Allora vale la seguente

Proposizione 6.16. Per ogni $j = 0, ..., m$ risulta $r_j = s_j r_0 + t_j r_1$.

Dimostrazione. Procediamo per induzione su j. La tesi è banalmente vera per $j = 0$ e per $j = 1$. Ora, si supponga vera la tesi per $j \leq i - 1$ e la si dimostri per $j = i$. Siccome, $r_{i-2} = q_{i-1} r_{i-1} + r_i$, sfruttando l'ipotesi induttiva si ha:

$$\begin{aligned} r_i &= r_{i-2} - q_{i-1} r_{i-1} = (s_{i-2} r_0 + t_{i-2} r_1) - q_{i-1} (s_{i-1} r_0 + t_{i-1} r_1) \\ &= (s_{i-2} - q_{i-1} s_{i-1}) r_0 + (t_{i-2} - q_{i-1} t_{i-1}) r_1 = s_i r_0 + t_i r_1, \end{aligned}$$

che è l'asserto.

□

Algoritmo 6.17. **(Algoritmo Euclideo Esteso (a, b))**

$a_0 \longleftarrow a$
$b_0 \longleftarrow b$
$t_0 \longleftarrow 0$
$t \longleftarrow 1$
$s_0 \longleftarrow 1$
$s \longleftarrow 0$
$q \longleftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
$r \longleftarrow a_0 - q b_0$
while $r > 0$
do
$temp \longleftarrow t_0 - q t_0$
$t_0 \longleftarrow t$
$t \longleftarrow temp$
$s_0 \longleftarrow s$
$s \longleftarrow temp$
$a_0 \longleftarrow b_0$
$b_0 \longleftarrow r$
$q \longleftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
$r \longleftarrow a_0 - q b_0$
return (r, s, t)
comment $r = \gcd(a, b)$ and $r = sa + tb$

Corollario 6.18. *Se* $\gcd(r_0, r_1) = 1$*, allora* t_m *è l'inverso di* r_1 *modulo* r_0*.*

Dimostrazione. Dalla Proposizione precedente, vale che

$$s_m r_0 + t_m r_1 = r_m = 1,$$

essendo $r_m = \gcd(r_0, r_1)$ come conseguenza dell'algoritmo euclideo.
Quindi $t_m r_1 \equiv 1 \ mod\ r_0$, ovvero la tesi.

□

Algoritmo 6.19. **(Inverso moltiplicativo** (a, b)**)**

$a_0 \longleftarrow a$
$b_0 \longleftarrow b$
$t_0 \longleftarrow 0$
$t \longleftarrow 1$
$q \longleftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
$r \longleftarrow a_0 - qb_0$
while $r > 0$
do
- $temp \longleftarrow (t_0 - qt_0)\ mod\ a$
- $t_0 \longleftarrow t$
- $t \longleftarrow temp$
- $a_0 \longleftarrow b_0$
- $b_0 \longleftarrow r$
- $q \longleftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
- $r \longleftarrow a_0 - qb_0$

if $r \neq 1$
then b has no inverse modulo n
else return (t)

Corollario 6.20. *Se* $\gcd(r_0, r_1) = 1$*, allora l'inverso di* r_1 *modulo* r_0 *è determinato in* $O(\ln^3 a)$ *operazioni bit.*

Dimostrazione. Per il calcolo dei r_j,q_j,t_j e s_j vengono eseguite $O(\ln^2 a)$ operazioni bit, $O(\ln a)$ volte. Pertanto, per il calcolo t_m vengono eseguite $O(\ln^3 a)$ operazioni bit.

□

Esempio 6.21. Vogliamo determinare l'inverso di 28 modulo 75.
Posto $r_0 = 75$ e $r_1 = 28$, vale che $75 = 2 * 28 + 19$. Quindi $q_1 = 2$ e r_2

i	r_i	q_i	s_i	t_i
0	75	–	1	0
1	28	2	0	1
2	19	1	1	-2
3	9	2	-1	3
4	1	9	3	-8

Pertanto $3 * 75 + (-8) * 28 = 1$ e quindi $(-8) * 28 \equiv 1 \ mod\ 75$. Siccome $-8 \equiv 67 \ mod\ 75$, vale che 67 è l'inverso di 28 mod 75.

□

6.3.4 Complessità delle operazioni in $\mathbb{Z}_n$

In questo paragrafo forniamo il tempo impiegato per compiere le operazioni in $\mathbb{Z}_n$.

Sia k il numero delle cifre in binario di n e siano a, b due interi tali che $0 \leq a, b \leq n-1$.

- $T((a \pm b) \ \textbf{mod}\ n) = O(k)$**.**

  Proviamo solo che Tempo$((a \pm b) mod\ n) = O(k)$ (la differenza per esercizio). Siccome $0 \leq a, b \leq n-1$, allora $0 \leq a+b \leq 2n-2$. Quindi se $a+b \leq n-1$, allora $(a+b) mod\ n = a+b$, se $a+b \geq n$, allora $(a+b) mod\ n = a+b-n$. Quindi in ogni caso vengono impiegate $O(k)$ bit operazioni.

- $T(ab \ \textbf{mod}\ n) = O(k^2)$.

  Per il calcolo di ab vengono impiegate al più k^2 bit operazioni. Inoltre ab è un numero di $2k$ bit quindi la successiva divisione per k impiega $4k^2$ bit. Quindi il calcolo di $ab \ mod\ n$ impiega $O(k^2)$ bit.

- **Se** $\gcd(a, n) = 1$**, allora** $T(a^{-1} \ \textbf{mod}\ n) = O(k^3)$**.**

  Vero per il **Corollario 6.20**.

- $T(a^b \ \textbf{mod}\ n) = O(k^2 \log b)$**.**

  In particolare, se $b < n$**, allora** $T(a^b \ \textbf{mod}\ n) = O(k^3)$**.**

  Sia $b = \sum_{i=0}^{\ell-1} b_i 2^i$ con $b_i = 0, 1$, allora il calcolo di $a^b \ mod\ n$ avviene attraverso il Metodo dei Quadrati Ripetuti.

Algoritmo 6.22. **(Quadrati Ripetuti** (a, b, n)**)**

$z \longleftarrow 1$
for $i \longleftarrow \ell - 1$ **downto** 0
do { $z \longleftarrow z^2 \ mod \ n$
if $b_i = 1$
then $z \longleftarrow (z \times a) \ mod \ n$
return (z)

Siccome ad ogni passo si valuta $z^2 \ mod \ n$ e poi, eventualmente, $(z \times a) \ mod \ n$ vengono utilizzati $O(k^2)$ bit poichè $z^2 \ mod \ n$ è costituito da al più k bit.
I due prodotti $z^2 \ mod \ n$ e $(z \times a) \ mod \ n$ sono ripetuti al più ℓ volte. Pertanto, $T(a^b \ mod \ n) = O(k^2 \log b)$. In particolare, se $b < n$, allora $T(a^b \ mod \ n) = O(k^3)$.

Esempio 6.23. Calcolare $a^b \ mod \ n$, dove $a = 9726$, $b = 3533$ e $n = 11413$.

Siccome $b = 2^0 + 2^2 + 2^3 + 2^6 + 2^7 + 2^8 + 2^{10} + 2^{11}$, allora

i	b_i	$z \ mod \ n$
11	1	$1^2 \times 9726 = 9726$
10	1	$9726^2 \times 9726 = 2559$
9	0	$2559^2 = 5634$
8	1	$5634^2 \times 9726 = 9167$
7	1	$9167^2 \times 9726 = 4958$
6	1	$4958^2 \times 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \times 9726 = 10185$
2	1	$10185^2 \times 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \times 9726 = 5761$.

Pertanto, $9726^{3533} \ mod \ 11413 = 5761$.

□

7 Il Crittosistema RSA

7.1 Elementi di Teoria dei Numeri

Prima di definire il Crittosistema RSA, richiamiamo alcuni fatti elementari di Teoria dei Numeri.

Teorema 7.1. ***(Teorema Cinese dei Resti)***
Siano a_i, m_i, $i = 1, \dots, r$, interi con $gcd(m_i, m_j) = 1$ per ogni $i, j = 1, \dots, r$, $i \neq j$, e si consideri il seguente sistema congruenziale:

$$\mathcal{S}: \begin{cases} x \equiv a_1 \ mod \ m_1 \\ \vdots \\ x \equiv a_r \ mod \ m_r. \end{cases}$$

Allora risulta che:
(1) *il sistema $\mathcal{S}$ ammette soluzione;*
(2) *se x_1 e x_2 sono due soluzioni di $\mathcal{S}$, allora $x_1 \equiv x_2 \ mod \ M$, dove $M = \prod_{i=1}^{r} m_i$.*

Dimostrazione.
(1) Sia $M_i := \frac{M}{m_i}$ per ogni $i = 1, \dots, r$, allora si ha che $gcd(M_i, m_i) = 1$, poiché $gcd(m_i, m_j) = 1$ per ogni $i, j = 1, \dots, r$, $i \neq j$. Quindi esiste N_i tale che $M_i N_i \equiv 1 \ mod \ m_i$. Sia

$$x_0 := \sum_{h=1}^{r} a_h M_h N_h.$$

Risulta

$$m_i \mid M_h \text{ per } h \neq i \Rightarrow m_i \mid \sum_{h=1, h\neq i}^{r} a_h M_h N_h,$$

e quindi

$$x_0 \equiv a_i M_i N_i \ mod \ m_i.$$

Siccome $M_i N_i \equiv 1 \ mod \ m_i$, si ha $x_0 \equiv a_i \ mod \ m_i$ per ogni $i = 1, \dots, r$.
(2) Osserviamo che se x_1 e x_2 sono due soluzioni di $\mathcal{S}$, allora $x_1 \equiv x_2 \ mod \ m_i$ per ogni $i = 1, \dots, r$, e quindi, poiché $gcd(m_i, m_j) = 1$ per $i \neq j$, segue che $x_1 \equiv x_2 \ mod \ M$.

□

Corollario 7.2. *Il sistema $\mathcal{S}$ ammette un'unica soluzione compresa tra 0 ed $M - 1$.*

Dimostrazione. Dal **Teorema Cinese dei Resti**, le soluzioni di $\mathcal{S}$ sono tutti e soli gli elementi della classe di resto modulo M individuata da x_0. Poiché tale classe ha un unico rappresentante compreso tra 0 ed $M - 1$, segue l'asserto.

□

Definizione 7.3. Sia n un intero, allora

$$\varphi(n) = |\{1 \leq x \leq n : \gcd(x,n) = 1\}|$$

è la **"Phi" di Eulero** di n.

Sono fatti noti i seguenti:

Proposizione 7.4. Valgono i seguenti fatti:

(1) Se $p \in \mathbb{P}$ e $\alpha \in \mathbb{N}$, allora $\varphi(p^\alpha) = p^{\alpha-1}(p-1)$.

(2) Se $\gcd(n,m) = 1$, allora $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$.

(3) Se $n = \prod_{i=1}^{k} p_i^{\alpha_i}$, allora $\varphi(n) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p-1)$.

Dimostrazione.

(1) Per provare l'asserto **(1)** è sufficiente notare che i numeri naturali divisibili per p e compresi tra 1 e p^α sono tutti e soli quelli della forma xp con $x = 1, \ldots, p^{\alpha-1}$, che sono in totale $p^{\alpha-1}$.

(2) Per provare l'asserto **(2)**, si considerino i seguenti insiemi:

$$\begin{aligned} X(mn) &= \{b = 1, \ldots, mn-1 : gcd(b, mn) = 1\} \\ X(m) &= \{x = 1, \ldots, m-1 : gcd(x, m) = 1\} \\ X(n) &= \{y = 1, \ldots, n-1 : gcd(y, n) = 1\} \end{aligned}$$

e si consideri, inoltre, l'applicazione

$$f : X(mn) \longrightarrow X(m) \times X(n), b \longmapsto (b_1, b_2),$$

dove $b_1 \equiv b \ mod \ m$ e $b_2 \equiv b \ mod \ n$ con $b_1 = 1, \ldots, m-1$ e $b_2 = 1, \ldots, n-1$. Ovviamente risulta $gcd(b_1, m) = gcd(b_2, n) = 1$. Per il **Teorema Cinese dei Resti**, il sistema

$$\begin{cases} b \ mod \ m = b_1 \\ b \ mod \ n = b_2 \end{cases}$$

essendo $gcd(m,n) = 1$, ammette sempre un'unica soluzione compresa tra 0 e $mn-1$, e questo implica che f è una biiezione. Quindi

$$\varphi(nm) = |X(mn)| = |X(m)||X(n)| = \varphi(n)\varphi(m).$$

(3) Infine, sia $n = \prod_{i=1}^{k} p_i^{a_i}$ con p_h, p_j primi distinti per $h, j = 1, \ldots, k$, $h \neq j$. Da **(1)** e **(2)** segue che

$$\varphi(n) = \prod_{i=1}^{k} \varphi(p_i^{a_i}) = \prod_{i=1}^{k} (p_i^{a_i} - p_i^{a_i - 1}) = \prod_{i=1}^{k} p_i^{a_i} \left(1 - \frac{1}{p_i}\right).$$

Pertanto, anche l'asserto **(3)** è dimostrato.

□

Teorema 7.5. ***(Teorema di Eulero).***
Siano n, x interi tali che $\gcd(x, n) = 1$, allora $x^{\varphi(n)} \equiv 1 \ mod\ n$.

Dimostrazione. Siccome l'ordine di $\mathbb{Z}_n^*$ è $\varphi(n)$, segue dal **Teorema di Lagrange** che

$$x^{\varphi(n)} \equiv 1 \ mod\ n.$$

□

Corollario 7.6. ***(Piccolo Teorema di Fermat).***
Siano p un primo e x un intero tale che $p \nmid x$, allora $x^{p-1} \equiv 1 \ mod\ p$.

Dimostrazione. Segue dal **Teorema di Eulero**, osservando che $\varphi(p) = p - 1$.

□

7.2 Il Crittosistema RSA

Il Crittosistema RSA è un cifrario inventato da **Rivest**, **Adleman** e **Shamir** nel 1977. E' uno dei cifrari più usati ad oggi e fonda la sua sicurezza sulla difficoltà computazionale della fattorizzazione di interi costituiti da un numero elevato di cifre.

Figura 7.1: Ronald Linn Rivest (1947), Leonard Max Adleman (1945) e Adi Shamir (1952)

Il Crittosistema RSA è definito come segue:

Definizione 7.7. **(Crittosistema RSA)**
Sia $n = pq$, con p, q primi distinti, siano $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ e sia

$$\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1 \ mod \ \varphi(n)\}.$$

Infine, se $K \in \mathcal{K}$ allora

$$\begin{aligned} e_K &: \mathbb{Z}_n \longrightarrow \mathbb{Z}_n, x \longmapsto x^b \ mod \ n \\ d_K &: \mathbb{Z}_n \longrightarrow \mathbb{Z}_n, y \longmapsto y^a \ mod \ n \end{aligned}$$

La coppia (n, b) è detta **chiave pubblica**, la terna (p, q, a) è detta **chiave privata**.

Proviamo che di fatto l'RSA è un crittosistema. Come vedremo, questo è vero per il **Teorema di Eulero**.

Teorema 7.8. $d_K \circ e_K = Id_{\mathbb{Z}_n}$.

Dimostrazione. Analizziamo i casi $x \in \mathbb{Z}_n^*$ e $x \in \mathbb{Z}_n - \mathbb{Z}_n^*$ separatamente.

- **Supponiamo che** $x \in \mathbb{Z}_n^*$.
  Poiché $ab \equiv 1 \ mod \ \varphi(n)$, allora $ab = 1 + t\varphi(n)$ con $t \in \mathbb{Z}$. Quindi
  $$(x^b)^a = x^{ab} \equiv \left(x^{\varphi(n)}\right)^t x \ mod \ n \equiv \quad 1^t x \ mod \ n \equiv x \ mod \ n$$
  e quindi $d_K(e_K(x)) = x$ per $x \in \mathbb{Z}_n^*$.
- **Supponiamo che** $x \in \mathbb{Z}_n - \mathbb{Z}_n^*$.
  Se $x = 0$, la tesi è banalmente vera. Quindi, supponiamo che $0 < x \leq n - 1$. Se $\gcd(x, n) = p$, allora $\gcd(x, q) = 1$ e quindi $x^{q-1} \equiv 1 \ mod \ q$ per il **Piccolo Teorema di Fermat**. Allora
  $$x^{\varphi(n)} \equiv 1 \ mod \ q$$
  essendo $\varphi(n) = (p-1)(q-1)$. Quindi,
  $$(x^b)^a = x^{ab} = \left(x^{\varphi(n)}\right)^t x \equiv 1^t x \ mod \ q \equiv x \ mod \ q.$$
  D'altra parte, è banalmente vero che
  $$(x^b)^a \equiv 0 \ mod \ p \equiv x \ mod \ p$$
  siccome $p \mid x$. Quindi
  $$\begin{cases} x^{ab} \equiv x \ mod \ q \\ x^{ab} \equiv x \ mod \ p \end{cases}$$
  Siccome $p \neq q$ segue che $(x^{ab}) \equiv x \ mod \ n$. Cioè, $d_K(e_K(x)) = x$ per $x \in \mathbb{Z}_n - \mathbb{Z}_n^*$. Il caso $\gcd(x, n) = q$ è dimostrato in maniera analoga.

□

Esempio 7.9. L'utente B sceglie i primi $p = 101$ e $q = 113$. Allora $n = 11413$ e

$$\varphi(n) = (101-1)(113-1) = 11200 = 2^6 5^2 7,$$

sceglie un intero b primo con $\varphi(n)$, ovvero non divisibile per $2, 5$ e 7, per esempio $b = 3533$ allora b è invertibile in $\mathbb{Z}_{11200}$ e $a = b^{-1} \ mod\ 11200$ è 6597. Quindi la chiave pubblica dell'utente B è $(11413, 3533)$, quella privata è $(101, 113, 6597)$. Supponiamo che l'utente A voglia trasmettere l'unità di messaggio in chiaro $x = 9726$. Allora calcola

$$y = 9726^{3533} \ mod\ 11413 = 5761$$

e lo trasmette a B che una volta ricevuto calcola

$$x = 5761^{6597} \ mod\ 11413 = 9726.$$

(i) La sicurezza dell'RSA consiste nell'**assunto** che la funzione $e_K(x) = x^b \ mod\ n$ sia **trapdoor**, **one-way**. Quindi, è computazionalmente difficile invertirla se non si hanno delle informazioni aggiuntive.

(ii) La **trapdoor** che permette al destinatario prestabilito di decifrare un testo cifrato è la conoscenza della fattorizzazione di $n = pq$. Infatti, esso calcola immediatamente $\varphi(n) = (p-1)(q-1)$ e determina b, l'inverso di a modulo $\varphi(n)$ utilizzando l'algoritmo di euclideo esteso.

7.2.1 Implementare l'RSA

Inseguito analizzeremo i seguenti aspetti del crittosistema RSA:

I. Costruzione del crittosistema.

II. Efficienza della cifratura e della decifratura.

III. Il problema della sicurezza.

L'efficienza della cifratura e della decifratura è data dal fatto che il calcolo di $e_K(x) = x^b \ mod\ n$ e $d_K(y) = y^a \ mod\ n$, noti b e a, hanno entrambi complessità $O(\log^3 n)$. Quindi, rimangono da analizzare gli aspetti **I.** e **III.**

La costruzione del crittosistema RSA avviene attraverso un algoritmo che determina i parametri dell'RSA e opera come segue:

1. Genesi di due primi distinti p, q.
2. Calcolo di $n = pq$ e di $\varphi(n) = (p-1)(q-1)$.
3. Scelta di un intero casuale $1 < b < \varphi(n)$ tale che $\gcd(b, \varphi(n)) = 1$.
4. Calcolo dell'inverso a di b modulo $\varphi(n)$.
5. La chiave pubblica è (n, b), quella segreta è (p, q, a).

- I primi p, q devono essere grandi, altrimenti un attacco ovvio all'RSA consiste nella fattorizzazione di n (è stato congetturato che che violare l'RSA è polinomialmente equivalente alla fattorizzazione di n). In generale, è sufficiente che p, q siano interi di almeno 512 bit. Pertanto, n ha almeno 1024 bit e al momento non esistono algoritmi efficienti per la fattorizzazione di tali interi. Tralasciando per il momento lo step **(1)** che verrà analizzato in seguito in dettaglio e che si prova avere complessità $O(\log^3 n)$.
- Il calcolo dello n e $\varphi(n)$ ha complessità totale $O(\log^2 n)$.
- Per gli step **(3)** e **(4)** si utilizza l'algoritmo euclideo che ha complessità $O(\log^2 n)$.

Pertanto, l'algoritmo che sta alla base della costruzione dell'RSA ha complessità $O(\log^3 n)$ che è una funzione polinomiale del numero dei bit in un unità di messaggio in chiaro (o cifrato).

Se da una parte il crittosistema RSA è dimostrabilmente sicuro, dall'altra, come tutti i crittosistemi a chiave pubblica, non è incondizionatamente sicuro.
Infatti, un avversario osservando un testo cifrato y ed avendo infinite risorse computazionali valuta $e_K(x)$ al variare di x in $\mathbb{Z}_n$ fino a quando non ottiene $e_K(x_0) = y$ e quindi decifra y con x_0.

Quindi, in seguito analizzeremo la sicurezza computazionale dell'RSA.

8 Test di Primalità

Nella costruzione del crittosistema RSA è necessario generare due numeri primi 'casuali'.

Nella pratica si generano interi dispari casuali e si testa la loro primalità attraverso gli algoritmi di **Soloway-Strassen** e **Miller-Rabin**. Entrambi, come vedremo, sono algoritmi Montecarlo randomizzati il cui tempo di esecuzione è polinomiale.

Per comprendere siffatti test è necessario introdurre ulteriori nozioni di Teoria dei Numeri. Ricordiamo i seguenti risultati:

1. Per ogni p^m, p primo, $m \in \mathbb{N}$, esiste, a meno di un isomorfismo, un unico campo finito $GF(p^m)$ di ordine p^m. (Esso si ottiene come campo di spezzamento del polinomio $X^{p^m} - X \in \mathbb{Z}_p[X]$);
2. Per il **Teorema di Cauchy** $GF(p^m)^*$, l'insieme degli elementi non nulli di $GF(p^m)$, è un gruppo ciclico di ordine $p^m - 1$;
3. Il numero dei generatori di $GF(p^m)^*$ è $\varphi(p^m - 1)$;
4. I sottocampi di $GF(p^m)$ sono tutti e soli i campi $GF(p^i)$ con $i \mid m$.

Definizione 8.1. **(Radice n-esima dell'unità)**
Un elemento x di $GF(q)^*$, $q = p^m$, si dice **radice n-esima dell'unità**, se $x^n = 1$. Se, inoltre, le radici n-esime dell'unità sono tutte e sole le potenze di x, allora x si dice **radice n-esima primitiva dell'unità**.

Proposizione 8.2. Valgono i seguenti risultati:

- **(1)** Sia g un generatore di $GF(q)^*$, allora g^j è una radice n-esima dell'unità se e solo se $nj \equiv_{q-1} 0$;
- **(2)** Il numero delle radici n-esime dell'unità è $gcd(n, q-1)$;
- **(3)** $GF(q)$ ha radici n-esime primitive dell'unità se e solo se $n \mid q-1$;
- **(4)** Se x_0 è una radice n-esima primitiva dell'unità in $GF(q)$, allora x_0^j è una radice n-esima primitiva dell'unità se e solo se $gcd(j, n) = 1$.

Dimostrazione.

(1) Sia g un generatore di $GF(q)^*$. Allora $o(g) = q-1$, e quindi g^j è una radice n-esima dell'unità se e solo se $g^{jn} = 1$, cioè $q-1 \mid nj$ e quindi l'asserto **(1)** è dimostrato.

(2)-(3) Il numero delle radici n-esime dell'unità è uguale al numero degli j compresi tra 1 e $q-1$ tali che $(q-1) \mid nj$. Posto $d = gcd(n, q-1)$, si ha $\frac{q-1}{d} \mid \frac{n}{d} j$ e poiché $gcd\left(\frac{q-1}{d}, \frac{n}{d}\right) = 1$, risulta: $\frac{q-1}{d} \mid j$. Quindi le radici n-esime dell'unità sono tutti e soli gli elementi della forma $g^{\frac{q-1}{d}k}$ con $k = 1, \ldots, d$ che sono appunto d. Osserviamo inoltre che tali radici sono n se e solo se $d = n$, i.e. se $n \mid q-1$. Pertanto valgono gli asserti **(2)** e **(3)**.

(4) Infine, sia x_0 radice n-esima primitiva dell'unità, allora $\mathcal{R} = \left\{x_0^i : i = 1, \ldots, n\right\}$ rappresenta l'insieme di tutte le radici n-esime dell'unità. $\mathcal{R}$ è il sottogruppo di $GF(q)^*$ di ordine n. Allora x_0^j è una radice n-esima primitiva dell'unità se e solo se x_0^j è un generatore di $\mathcal{R}$ e ciò si verifica se e solo se $gcd(j, n) = 1$. Abbiamo così provato l'asserto **(4)**.

□

Sia p un primo dispari e sia $a \in GF(p)^*$. Allora a è un quadrato in $GF(p)$ se e solo se esiste b in $GF(p)$ tale che $a = b^2$. In tal caso, a ha precisamente due radici quadrate: $\pm b$. Pertanto i quadrati in $GF(p)^*$ possono essere trovati calcolando $b^2 \ mod \ p$ per $b = 1, 2, 3, \ldots, \frac{(p-1)}{2}$ (poiché i restanti interi fino a $p-1$ sono tutti congrui a $-b$ per ciascuno di tali b), e precisamente la metà degli elementi in $GF(p)^*$ sono quadrati.

Definizione 8.3. **(Residui Quadratici modulo p)**
I quadrati in $GF(p)$ sono detti **residui quadratici modulo p**. I restanti elementi non nulli sono detti **residui non quadratici**.

Esempio 8.4. In $GF(11)$ ci sono 5 residui quadrati che sono: $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 5$ e $5^2 = 3$, e 5 non quadrati che sono: $2, 6, 7, 8, 10$.

Poiché il prodotto di due quadrati in $GF(p)^*$ e l'inverso di un quadrato in $GF(p)^*$ sono ancora dei quadrati, allora l'insieme Q dei quadrati di $GF(p)^*$ è un sottogruppo di $GF(p)^*$.

L'applicazione

$$\alpha : \ GF(p)^* \longrightarrow Q, \quad x \longmapsto x^2$$

è un omomorfismo suriettivo di gruppi e quindi $\frac{GF(p)^*}{ker\alpha} \cong Q$, dove $\ker \alpha$ è il sottogruppo di $GF(p)^*$ costituito dalle radici quadrate di 1. Poiché $\ker \alpha = \{\pm 1\}$, essendo p dispari, si ha $|Q| = \frac{p-1}{2}$.

E' facile verificare che valgono, inoltre, le seguenti proprietà:

1. il prodotto di due quadrati o di due non quadrati è un quadrato in $GF(p)^*$;
2. il prodotto di un quadrato e di un non quadrato è un non quadrato in $GF(p)^*$.

Proposizione 8.5. -1 è un quadrato in $GF(p)$ se e solo se $p \equiv_4 1$.

Dimostrazione. -1 è un quadrato in $GF(p)$ se e solo se $x^2 = -1$, i.e. $x^4 = 1$, cioè se e solo se $GF(p)$ ammette radici quarte primitive dell'unità. Per la **Proposizione 8.2 (3)**, ciò si verifica solo se $p \equiv_4 1$.

□

Definizione 8.6. **(Simbolo di Legendre)**
Siano p un primo dispari ed a un intero, allora:

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{se } p \mid a \\ 1 & \text{se } a \text{ è un residuo quadratico in } GF(p) \\ -1 & \text{se } a \text{ è un residuo non quadratico in } GF(p) \end{cases}$$

$\left(\frac{a}{p}\right)$ è detto **Simbolo di Legendre**.

Teorema 8.7. ***(Teorema di Eulero)***
Per il Simbolo di Legendre vale la seguente relazione:

$$\left(\frac{a}{p}\right) \equiv_p a^{\frac{p-1}{2}}$$

Dimostrazione. Se p divide a, l'asserto segue banalmente.
Quindi supponiamo $(a,p) = 1$. Per il **Piccolo Teorema di Fermat** segue che $a^{p-1} \equiv_p 1$ e quindi $a^{\frac{p-1}{2}} \equiv_p \pm 1$, essendo p dispari. Sia g un generatore di $GF(p)^*$. Allora esiste un intero j tale che $a \equiv_p g^j$. Pertanto $a^{\frac{p-1}{2}} \equiv_p g^{j\frac{p-1}{2}}$. Osserviamo che $a^{\frac{p-1}{2}} \equiv_p g^{j\frac{p-1}{2}} \equiv_p 1$ se e solo se $p-1$ divide $j\frac{p-1}{2}$. Ciò si verifica se e solo se j è pari e quindi se e solo se a è un residuo quadratico in $GF(p)$, i.e. $\left(\frac{a}{p}\right) = 1$.

□

Proposizione 8.8. Il Simbolo di Legendre soddisfa le seguenti proprietà:

(1) Se $a_0 \equiv_p a$, allora $\left(\frac{a}{p}\right) = \left(\frac{a_0}{p}\right)$;

(2) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$;

(3) Se $(b,p) = 1$, allora $\left(\frac{ab^2}{p}\right) = \left(\frac{a}{p}\right)$;

(4) $\left(\frac{1}{p}\right) = 1$ e $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$.

Dimostrazione.

(1) L'asserto **(1)** segue dalla definizione.

(2) Dal **Teorema 8.7** si ha

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \equiv_p \left(\frac{ab}{p}\right)$$

e quindi p divide $\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) - \left(\frac{ab}{p}\right)$. Inoltre p è dispari e poiché

$$\left|\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) - \left(\frac{ab}{p}\right)\right| \leq 2,$$

vale l'asserto **(2)**.

(3) L'asserto **(3)** segue da **(2)**.

(4) Infine, poiché risulta $\left|\left(\frac{-1}{p}\right) - (-1)^{\frac{p-1}{2}}\right| \leq 2$, con $\left(\frac{-1}{p}\right) - (-1)^{\frac{p-1}{2}}$ divisibile per p per il **Teorema 8.7**, vale l'asserto **(4)**.

□

Lemma 8.9. *Siano $n \in \mathbb{N}$ e*

$$f(n) = \begin{cases} (-1)^{\frac{n^2-1}{8}} & \text{se } n \text{ è dispari} \\ 0 & \text{se } n \text{ è pari} \end{cases}$$

allora $f(n_1n_2) = f(n_1)f(n_2)$ per ogni $n_1, n_2 \in \mathbb{N}$.

Dimostrazione. Siano $n_1, n_2 \in \mathbb{N}$. Se almeno uno tra n_1 ed n_2 è pari, l'asserto segue banalmente. Quindi si supponga che n_1 e n_2 siano entrambi dispari. Allora

$$\begin{aligned} f(n_1n_2) &= (-1)^{\frac{(n_1n_2)^2-1}{8}} = (-1)^{\frac{n_1^2n_2^2-n_1^2+n_1^2-1}{8}} = (-1)^{\frac{n_1^2(n_2^2-1)+n_1^2-1}{8}} \\ &= (-1)^{\frac{n_1^2-1}{8}}(-1)^{n_1^2\frac{n_2^2-1}{8}} = (-1)^{\frac{n_1^2-1}{8}}(-1)^{\frac{n_2^2-1}{8}} = f(n_1)f(n_2). \end{aligned}$$

□

Proposizione 8.10. Sia p un primo dispari. Allora:

$$\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} = \begin{cases} 1 & \text{se } p \equiv_8 \pm 1 \\ -1 & \text{se } p \equiv_8 \pm 3 \end{cases}$$

Dimostrazione. Poiché p è dispari, allora $8 \mid p^2 - 1$ e $GF(p^2)$ ammette radici ottave primitive dell'unità per la **Proposizione 8.2 (3)**. Sia ξ una di esse. Si definisca

$$H = \sum_{j=0}^{7} f(j)\xi^j,$$

dove $f(j) = (-1)^{\frac{j^2-1}{8}}$ se j è dispari e 0 altrimenti. Allora $H = \xi - \xi^3 - \xi^5 + \xi^7$. Siccome $\xi^4 = -1$, allora risulta $\xi^5 = -\xi$, $\xi^7 = -\xi^3$ e quindi $H = 2(\xi - \xi^3)$ e $H^2 = 4(\xi^2 - 2\xi^4 + \xi^6) = 8$. In particolare, $H \neq 0$. Dal **Teorema 8.7** e dalla **Proposizione 8.8 (3)**, tenendo presente che p è dispari, segue

$$H^p = (H^2)^{\frac{p-1}{2}} H = 8^{\frac{p-1}{2}} H = \left(\frac{8}{p}\right) H = \left(\frac{2}{p}\right) H. \tag{8.1}$$

Per il **Lemma 8.9**, vale che $f(j) = f(p)^2 f(j) = f(p)f(pj)$ e quindi

$$\begin{aligned} H^p &= \sum_{j=0}^{7} f(j)^p \xi^{jp} = \sum_{j=0}^{7} f(j)\xi^{jp} = \sum_{j=0}^{7} f(p)f(pj)\xi^{jp} \\ &= f(p) \sum_{j=0}^{7} f(pj)\xi^{jp}. \end{aligned}$$

L'insieme $\{jp \ : \ j = 0, \ldots, 7\}$ è un sistema completo di residui *mod* 8. Quindi per ogni $j = 0, \ldots, 7$ esiste un unico $j' = 0, \ldots, 7$ t.c. $jp \equiv_8 j'$. Pertanto $jp = j' + 8k$, con k intero. Allora $\xi^{jp} = \xi^{j'}\xi^{8k}$ e quindi $\xi^{jp} = \xi^{j'}$, essendo ξ una radice ottava dell'unità.

Inoltre vale che

$$f(jp) = f(j').$$

Infatti, j e j' appartengono alla stessa classe di parità, quindi $f(jp) = 0 = f(j')$ se j' è pari. Se j' è dispari, vale che

$$\begin{aligned} f(jp) = f(j' + 8k) &= (-1)^{\frac{(j'+8k)^2-1}{8}} = (-1)^{\frac{j'^2+16j'k+64k^2-1}{8}} = \\ &= (-1)^{\frac{j'^2-1}{8}} (-1)^{\frac{16j'k+64k^2}{8}} = f(j'). \end{aligned}$$

Pertanto, si ha che

$$H^p = f(p) \sum_{j=0}^{7} f(pj)\xi^{jp} = f(p) \sum_{j'=0}^{7} f(j')\xi^{j'} = f(p)H = (-1)^{\frac{p^2-1}{8}} H. \tag{8.2}$$

Ora uguagliando (8.1) e (8.2) e tenendo presente che $H \neq 0$, si ottiene l'asserto.

□

Definizione 8.11. **(Somma Gaussiana)**
Se p, q sono due primi dispari distinti tali che $\xi \in GF(p^f)$ è una radice q-esima primitiva dell'unità, allora

$$G = \sum_{j=0}^{q-1} \left(\frac{j}{q}\right) \xi^j$$

si dice **somma Gaussiana**.

Si noti che i primi della **Definizione 8.11** esistono. Infatti se f è un multiplo di $q-1$, allora $p^f \equiv_q 1$ per il **Piccolo Teorema di Fermat** e quindi le radici q–esime primitive dell'unità esistono per la **Proposizione 8.2 (3)**.

Lemma 8.12.

$$G^2 = (-1)^{\frac{q-1}{2}} q.$$

Dimostrazione. Poiché $\{-k : k = 0, \ldots, q-1\}$ è un sistema completo di residui modulo q, vale che per ogni $j = 0, \ldots, q-1$ esiste un unico $k = 0, \ldots, q-1$ tale che $j = -k + \vartheta q$, con $\vartheta \in \mathbb{Z}$. Allora $\left(\frac{j}{q}\right) = \left(\frac{-k}{q}\right)$ per la **Proposizione 8.8 (1)** e $\xi^j = \xi^{-k} \left(\xi^q\right)^\vartheta = \xi^{-k}$ essendo ξ una radice q–esima dell'unità. Quindi si ha che:

$$G = \sum_{j=0}^{q-1} \left(\frac{j}{q}\right) \xi^j = \sum_{k=0}^{q-1} \left(\frac{-k}{q}\right) \xi^{-k}.$$

Poiché $\left(\frac{0}{q}\right) = 0$, allora

$$\begin{aligned} G^2 &= \sum_{j=1}^{q-1} \left(\frac{j}{q}\right) \xi^j \sum_{k=1}^{q-1} \left(\frac{-k}{q}\right) \xi^{-k} \underset{\text{Prop. } 8(2)}{=} \\ &= \left(\frac{-1}{q}\right) \sum_{j=1}^{q-1} \left(\frac{j}{q}\right) \xi^j \sum_{k=1}^{q-1} \left(\frac{k}{q}\right) \xi^{-k}. \end{aligned}$$

Per un qualsiasi j fissato, l'insieme $\{jh : h = 0, \ldots, q-1\}$ è ancora un sistema completo di residui modulo q. Quindi per ogni $k = 0, \ldots, q-1$ esiste un unico $h \in \{0, \ldots, q-1\}$ tale che $k = jh + aq$, con $a \in \mathbb{Z}$. Allora, ragionando come sopra, valgono $\left(\frac{k}{q}\right) = \left(\frac{jh}{q}\right)$ e $\xi^{-k} = \xi^{-jh}$ e quindi

$$\sum_{k=1}^{q-1} \left(\frac{k}{q}\right) \xi^{-k} = \sum_{h=1}^{q-1} \left(\frac{jh}{q}\right) \xi^{-jh}.$$

Quindi

$$\begin{aligned} G^2 &= (-1)^{\frac{q-1}{2}} \sum_{j=1}^{q-1} \left(\frac{j}{q}\right) \xi^j \sum_{h=1}^{q-1} \left(\frac{jh}{q}\right) \xi^{-jh} \\ &= (-1)^{\frac{q-1}{2}} \sum_{j=1}^{q-1} \sum_{h=1}^{q-1} \left(\frac{j^2 h}{q}\right) \xi^{j(1-h)} \\ &= (-1)^{\frac{q-1}{2}} \sum_{j=1}^{q-1} \sum_{h=1}^{q-1} \left(\frac{h}{q}\right) \xi^{j(1-h)}. \end{aligned} \tag{8.3}$$

Siccome i quadrati e i non quadrati di $GF(q)^*$ sono in egual numero pari a $\frac{q-1}{2}$, vale che $\sum_{k=1}^{q-1}\left(\frac{k}{q}\right) = 0$ e quindi non si altera la somma (8.3) se si aggiunge il termine relativo a $j = 0$. Pertanto segue che:

$$G^2 = (-1)^{\frac{q-1}{2}} \sum_{j=0}^{q-1}\sum_{k=1}^{q-1}\left(\frac{k}{q}\right)\xi^{j(1-k)} = (-1)^{\frac{q-1}{2}} \sum_{k=1}^{q-1}\left(\frac{k}{q}\right)\sum_{j=0}^{q-1}\xi^{j(1-k)}. \tag{8.4}$$

Se $k \neq 1$, si ha:

$$\sum_{j=0}^{q-1}\xi^{j(1-k)} = (\xi^{q(1-k)} - 1)(\xi^{(1-k)} - 1)^{-1} = 0$$

e in (8.4) resta solo l'addendo relativo a $k = 1$. Così $G^2 = (-1)^{\frac{q-1}{2}}\, q$.

□

Teorema 8.13. ***(Legge di Reciprocità Quadratica)***
Se p, q sono due primi dispari, allora

$$\left(\frac{q}{p}\right) = (-1)^{\frac{(p-1)(q-1)}{4}}\left(\frac{p}{q}\right) = \begin{cases} -\left(\frac{p}{q}\right) & \text{se } p, q \equiv_4 3, \\ \left(\frac{p}{q}\right) & \text{altrimenti.} \end{cases}$$

Dimostrazione. Sia f un qualsiasi intero positivo tale che $q \mid p^f - 1$. Per la nota relativa alla **Definizione 8.11**, tali f esistono. Allora ha senso considerare la somma Gaussiana in $GF(p^f)$

$$G = \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\xi^j.$$

Dal **Lemma 8.12** e dal **Teorema 8.7**, tenendo presente che le congruenze modulo p sono uguaglianze in $GF(p)$ e quindi in $GF(p^f)$, segue che:

$$G^p = (G^2)^{\frac{p-1}{2}} G = (-1)^{\frac{(p-1)(q-1)}{4}} q^{\frac{p-1}{2}} G = (-1)^{\frac{(p-1)(q-1)}{4}}\left(\frac{q}{p}\right) G.$$

D'altra parte

$$\begin{aligned} G^p &= \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)^p \xi^{jp} = \sum_{j=0}^{q-1}\left(\frac{j}{q}\right)\xi^{jp} = \\ &= \sum_{j=0}^{q-1}\left(\frac{p}{q}\right)\left(\frac{jp}{q}\right)\xi^{jp} = \left(\frac{p}{q}\right)\sum_{j=0}^{q-1}\left(\frac{jp}{q}\right)\xi^{jp} \end{aligned}$$

per la **Proposizione 8.8 (2)-(3)**. Poiché l'insieme $\{jp : j = 0, \ldots, q-1\}$ è un sistema completo di residui modulo q, allora per ogni $j = 0, \ldots, q-1$ esiste un unico $j' = 0, \ldots, q-1$ tale che valgono $\left(\frac{jp}{q}\right) = \left(\frac{j'}{q}\right)$ e $\xi^{jp} = \xi^{j'}$.

Pertanto si ha $\sum_{j=0}^{q-1}\left(\frac{jp}{q}\right)\xi^{jp} = \sum_{j'=0}^{q-1}\left(\frac{j'}{q}\right)\xi^{j'} = G$ e quindi

$$(-1)^{\frac{(p-1)(q-1)}{4}}\left(\frac{q}{p}\right)G = G^p = \left(\frac{p}{q}\right)G. \tag{8.5}$$

Poiché per il **Lemma 8.12** risulta $G \neq 0$, possiamo dividere per G in (8.5) ottenendo così la tesi.

□

Esempio 8.14. Dati i primi 7411 e 9283, stabiliamo se 7411 è un residuo quadratico in $GF(9283)$.

Poiché 7411 e 9283 sono entrambi congrui a 3 *mod* 4, allora vale che

$$\left(\frac{7411}{9283}\right) = -\left(\frac{9283}{7411}\right).$$

Risulta:

$$\begin{aligned}\left(\frac{7411}{9283}\right) &= -\left(\frac{9283}{7411}\right) = -\left(\frac{1872}{7411}\right) = -\left(\frac{2^4\cdot 3^2\cdot 13}{7411}\right) = \\ &= -\left(\frac{2^4}{7411}\right)\left(\frac{3^2}{7411}\right)\left(\frac{13}{7411}\right) = -\left(\frac{13}{7411}\right) = \\ &= -\left(\frac{7411}{13}\right) = -\left(\frac{1}{13}\right) = -1\end{aligned}$$

□

Definizione 8.15. **(Simbolo di Jacobi)**
Siano a un intero e n un intero positivo dispari. Se $n = \prod_{i=1}^{k} p_i^{\alpha_i}$, $p_i \in \mathbb{P}$ e $\alpha_i \in \mathbb{N}$, allora si definisce **Simbolo di Jacobi** $\left(\frac{a}{n}\right)$ il prodotto dei Simboli di Legendre dei fattori primi di n:

$$\left(\frac{a}{n}\right) := \prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{\alpha_i}.$$

Remark 8.16. Si noti che $\left(\frac{a}{n}\right) = 1$, con n composto, non implica che a è un quadrato modulo n. Per esempio $\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1) = 1$, mentre non esiste un intero x tale che $x^2 \equiv_{15} 2$. Infatti, $x^2 \equiv_{15} 2$ implica $x^2 \equiv_3 2$, mentre 2 è un non quadrato in $GF(3)$.

Proposizione 8.17. Siano a, b, n interi, con n dispari. Valgono le seguenti proprietà:

(1) $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$;

(2) Se $a \equiv_n b$, allora $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.

Dimostrazione. Sia $n = \prod_{i=1}^{k} p_i^{\alpha_i}$ con p_i primo.

(1) Dalla definizione di Simbolo di Jacobi e dalla **Proposizione 8.8 (2)**, segue:

$$\left(\frac{ab}{n}\right) = \prod_{i=1}^{k}\left(\frac{ab}{p_i}\right)^{\alpha_i} = \prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{\alpha_i}\left(\frac{b}{p_i}\right)^{\alpha_i} = \left[\prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{\alpha_i}\right]\left[\prod_{i=1}^{k}\left(\frac{b}{p_i}\right)^{\alpha_i}\right] = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right).$$

quindi vale l'asserto **(1)**.

(2) Se $a \equiv_n b$, allora $a \equiv_{p_i} b$ per ogni primo $i = 1, \cdots, k$, e quindi dalla **Proposizione 8.8(1)** segue che

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{\alpha_i} = \prod_{i=1}^{k}\left(\frac{b}{p_i}\right)^{\alpha_i} = \left(\frac{b}{n}\right).$$

Pertanto vale l'asserto **(2)**.

□

Proposizione 8.18. Per ogni $n \in \mathbb{N}$, n dispari, vale che:

$$\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}.$$

Dimostrazione. Se $n = \prod_{i=1}^{k} p_i^{\alpha_i}$, p_i primo, allora dalla **Proposizione 8.10** e dal **Lemma 8.9** segue:

$$\begin{aligned}\left(\frac{2}{n}\right) &= \prod_{i=1}^{k}\left(\frac{2}{p_i}\right)^{\alpha_i} = \prod_{i=1}^{k}\left((-1)^{\frac{p_i^2-1}{8}}\right)^{\alpha_i} = \prod_{i=1}^{k} f(p_i)^{\alpha_i} \\ &= \prod_{i=1}^{k} f(p_i^{\alpha_i}) = f\left(\prod_{i=1}^{k} p_i^{\alpha_i}\right) = f(n) = (-1)^{\frac{n^2-1}{8}}.\end{aligned}$$

□

Proposizione 8.19. Per ogni $n, m \in \mathbb{N}$, n, m dispari, vale che:

$$\left(\frac{m}{n}\right) = (-1)^{\frac{(n-1)(m-1)}{4}} \left(\frac{n}{m}\right).$$

Dimostrazione. Supponiamo che $n = \prod_{i=1}^{k} p_i$ e $m = \prod_{j=1}^{h} q_j$, con p_i, q_j primi. Se $(n, m) > 1$, dalla definizione di Simbolo di Jacobi, si ha $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right) = 0$.
Se $(n, m) = 1$, allora per la **Proposizione 8.17** si ha

$$\left(\frac{m}{n}\right) = \prod_{j=1}^{h} \left(\frac{q_j}{n}\right) = \prod_{i,j=1}^{k,h} \left(\frac{q_j}{p_i}\right).$$

Ora, applicando kh volte la **Legge di Reciprocità** quadratica si ha:

$$\begin{aligned}
\left(\frac{m}{n}\right) &= \prod_{j=1}^{k} \left(\frac{q_j}{n}\right) = \prod_{i,j=1}^{k,h} \left(\frac{q_j}{p_i}\right) = \prod_{i,j=1}^{k,h} (-1)^{\frac{(p_i-1)(q_j-1)}{4}} \left(\frac{p_i}{q_j}\right) = \\
&= \left(\prod_{i,j=1}^{k,h} (-1)^{\frac{(p_i-1)(q_j-1)}{4}}\right) \prod_{i,j=1}^{k,h} \left(\frac{p_i}{q_j}\right) = \\
&= \left(\prod_{i,j=1}^{k,h} (-1)^{\frac{(p_i-1)(q_j-1)}{4}}\right) \left(\frac{n}{m}\right).
\end{aligned}$$

Notiamo che $(-1)^{\frac{(p_i-1)(q_j-1)}{4}} = -1$ se e solo se $p_i \equiv_4 3$ e $q_j \equiv_4 3$. Ora, se $0 \leq k_0 \leq k$ e $0 \leq h_0 \leq h$ rappresentano il numero dei fattori primi di n ed m rispettivamente, eventualmente ripetuti, congrui a 3 modulo 4, allora $n \equiv_4 3^{k_0}$, $m \equiv_4 3^{h_0}$ e

$$\prod_{i,j=1}^{k,h} (-1)^{\frac{(p_i-1)(q_j-1)}{4}} = (-1)^{k_0 h_0}.$$

Osserviamo che $(-1)^{k_0 h_0} = -1$ se e solo se k_0 e h_0 sono entrambi dispari. Cioè se $n \equiv_4 3$ e $m \equiv_4 3$ e quindi $(-1)^{\frac{(n-1)(m-1)}{4}} = -1$. Ciò completa la dimostrazione.

□

Esempio 8.20. Calcoliamo $\left(\frac{7411}{9283}\right)$ utilizzando il Simbolo di Jacobi.

Per la **Proposizione 8.19** segue che $\left(\frac{7411}{9283}\right) = -\left(\frac{9283}{7411}\right) = -\left(\frac{1872}{7411}\right)$. Osserviamo che, a questo punto, nell'**Esempio 8.14**, era necessario fattorizzare il numero 1872 per poter utilizzare il simbolo di Legendre. In questo caso, invece, possiamo evitare tale fattorizzazione, infatti:

$$\begin{aligned}
-\left(\frac{1872}{7411}\right) &= -\left(\frac{7411}{1872}\right) = -\left(\frac{1795}{1872}\right) = -\left(\frac{1872}{1795}\right) = -\left(\frac{77}{1795}\right) = -\left(\frac{1795}{77}\right) = -\left(\frac{24}{77}\right) = \\
&= -\left(\frac{2^3 \cdot 3}{77}\right) = -\left(\frac{2^3}{77}\right)\left(\frac{3}{77}\right) = -\left(\frac{2}{77}\right)\left(\frac{3}{77}\right) = \left(\frac{3}{77}\right) = \left(\frac{77}{3}\right) = \left(\frac{2}{3}\right) = -1.
\end{aligned}$$

□

8.1 Pseudoprimi

Definizione 8.21. **(Pseudoprimo)**
Siano n un intero composto dispari e b un intero tale che $gcd(b, n) = 1$. Se

$$b^{n-1} \equiv_n 1,$$

allora n si dice **pseudoprimo** rispetto alla base b.

Esempio 8.22. L'intero $n = 91 = 7 \cdot 13$ é uno pseudoprimo rispetto alla base $b = 3$ ma non rispetto alla base $b = 2$. Infatti $3^{90} \equiv_{91} 1$ mentre $2^{90} \equiv_{91} 64$.

Si noti che se b é una base per n, allora é base per n ogni intero appartenente alla classe di resto modulo n individuata da b.

Poiché ogni classe di resto modulo n ha un unico rappresentante compreso tra 0 ed $n - 1$, nel seguito, a meno che non sia esplicitamente detto il contrario, quando parleremo di basi rispetto ad n, faremo riferimento esclusivamente a tali residui modulo n.

Si noti inoltre che le basi per n sono $\varphi(n)$ e sono tutti e soli gli elementi invertibili in $\mathbb{Z}_n$ rispetto al prodotto (i.e. $\mathcal{U}(\mathbb{Z}_n)$).

Definizione 8.23. **(Numero di Carmichael)**
Un intero composto dispari che é pseudoprimo rispetto a tutte le basi, si dice **numero di Carmichael**.

Proposizione 8.24. Valgono le seguenti proprietà:

1. Sia n un intero composto dispari, allora n é un numero di Carmichael se e solo se per ogni primo p divisore di n vale che $p^2 \nmid n$ ma $p-1 \mid n-1$.
2. Un numero di Carmichael é prodotto di almeno tre primi distinti e il più piccolo numero di Carmichael é 561.
3. I numeri di Carmichael sono infiniti (Teorema di *Alford, Granville, Pomerance*, 1994).

8.1.1 Pseudoprimi di Eulero

Definizione 8.25. **(Pseudoprimo di Eulero)**
Siano n un intero composto dispari e b un intero tale che $gcd(n,b) = 1$. Allora n si dice **pseudoprimo di Eulero** rispetto alla base b se vale

$$b^{\frac{n-1}{2}} \equiv_n \left(\frac{b}{n}\right), \tag{8.6}$$

dove $\left(\frac{b}{n}\right)$ è il simbolo di Jacobi.

Si noti che ogni primo dispari soddisfa (8.6) in virtù del **Teorema 8.7**.

Teorema 8.26. *Sia n un intero composto dispari e sia Y l'insieme delle basi rispetto alle quali n è uno pseudoprimo di Eulero. Allora*

$$|\mathcal{U}(\mathbb{Z}_n) - Y| \geq \frac{1}{2}\varphi(n).$$

Dimostrazione. Sia

$$Y = \left\{ b \in \mathcal{U}(\mathbb{Z}_n) : b^{\frac{n-1}{2}} \equiv_n \left(\frac{b}{n}\right) \right\}.$$

Proviamo che $|\mathcal{U}(\mathbb{Z}_n) - Y| > 0$.

Supponiamo che esista una base b' tale che $(b')^{n-1} \not\equiv_n 1$, allora $(b')^{\frac{n-1}{2}} \not\equiv \left(\frac{b}{n}\right)$ e quindi $|\mathcal{U}(\mathbb{Z}_n) - Y| > 0$.

Pertanto supponiamo che $b^{n-1} \equiv_n 1$ rispetto a tutte le basi. Allora vale che $gcd(p, n/p) = 1$ per ogni divisore primo p di n. Sia x un non quadrato in $GF(p)^*$. Per il **Teorema Cinese dei Resti** esiste b_0 tale che $b_0 \equiv_p x$ e $b_0 \equiv_{n/p} 1$.
Quindi $gcd(b_0, n) = 1$ e $(\frac{b_0}{p}) = -1$, $(\frac{b_0}{n/p}) = 1$. Allora $(\frac{b_0}{n}) = -1$. Se vale che $b_0^{\frac{n-1}{2}} \equiv_n -1$, allora $b_0^{\frac{n-1}{2}} \equiv_{n/p} -1$. Ciò è assurdo poiché $b_0 \equiv_{n/p} 1$ e $b_0^{\frac{n-1}{2}} \equiv_{n/p} 1$.

Quindi $b_0 \in \mathcal{U}(\mathbb{Z}_n) - Y$ e $|\mathcal{U}(\mathbb{Z}_n) - Y| > 0$. Proviamo che $|\mathcal{U}(\mathbb{Z}_n) - Y| \geq |Y|$.

Siano $b \in Y$ e $b_0 \in \mathcal{U}(\mathbb{Z}_n) - Y$. Allora $bb_0 \in Y$ oppure $bb_0 \in U(\mathbb{Z}_n) - Y$. Se $bb_0 \in Y$, allora $(bb_0)^{\frac{n-1}{2}} \equiv_n (\frac{bb_0}{n})$, da cui segue $b^{\frac{n-1}{2}} b_0^{\frac{n-1}{2}} \equiv_n (\frac{b}{n})(\frac{b_0}{n})$.

Pertanto $(\frac{b}{n})b_0^{\frac{n-1}{2}} \equiv_n (\frac{b}{n})(\frac{b_0}{n})$, da cui si ricava $b_0^{\frac{n-1}{2}} \equiv_n (\frac{b_0}{n})$ e quindi $b_0 \in Y$ ma ciò è assurdo. Pertanto $bb_0 \in \mathcal{U}(\mathbb{Z}_n) - Y$ per ogni $b \in Y$. Quindi segue che

$$|\mathcal{U}(\mathbb{Z}_n) - Y| \geq |Yb_0| = |Y|.$$

Ora

$$\varphi(n) = |\mathcal{U}(\mathbb{Z}_n)| = |Y| + |\mathcal{U}(\mathbb{Z}_n) - Y| \leq 2\,|\mathcal{U}(\mathbb{Z}_n) - Y|$$

da cui segue

$$|\mathcal{U}(\mathbb{Z}_n) - Y| \geq \frac{1}{2}\varphi(n).$$

□

Osservazione 8.27. La probabilità che un intero composto dispari n sia uno pseudoprimo di Eulero rispetto a k basi distinte è al più $\frac{1}{2^k}$.

Esempio 8.28. $n = 561 = 3 \times 11 \times 17$ non è uno pseudoprimo di Eulero.

Infatti, per $p = 17$, siccome $\left(\frac{3}{17}\right) = -1$ si consideri il sistema congruenziale

$$\begin{cases} b \equiv_{17} 3 \\ b \equiv_{33} 1 \end{cases}$$

Utilizzando il **Teorema Cinese dei Resti**, la soluzione compresa tra 0 e 560 è $b = 496$. Allora

$$\left(\frac{496}{561}\right) = \left(\frac{496}{33}\right)\left(\frac{496}{17}\right) = \left(\frac{1}{33}\right)\left(\frac{3}{17}\right) = -1$$

Se 561 fosse uno pseudoprimo di Eulero rispetto alla base $b = 496$, si dovrebbe avere $496^{280} \equiv_{561} -1$ e quindi $496^{280} \equiv_{33} -1$. Ciò è assurdo, perchè $496 \equiv_{33} 1$ implica $496^{280} \equiv_{33} 1$. Pertanto 561 non è uno pseudoprimo di Eulero rispetto alla base $b = 496$.

□

8.1.2 Pseudoprimi Forti

Definizione 8.29. **(Pseudoprimo Forte)**
Sia n un intero composto dispari, quindi $n = 2^s t + 1$ con t dispari ed $s \geq 1$, e sia b un intero tale che $gcd(b, n) = 1$. Allora n si dice **pseudoprimo forte rispetto alla base** b, se

- $b^t \equiv_n 1$, oppure
- esiste $0 \leq r < s$ tale che $b^{2^r t} \equiv_n -1$.

Osserviamo che nella **Definizione 8.29** viene escluso il caso $r = s$. Infatti, se fosse $r = s$, allora $b^{n-1} \equiv_n -1$. Quindi l'ordine di b, in quanto invertibile nelle classi di resto modulo n, dovrebbe essere $2(n-1)$. D'altra parte, l'ordine di b divide $\varphi(n)$ e $\varphi(n) < 2(n-1)$.

Ad esempio, sia $n = 65 = 2^6 + 1$.

- Consideriamo $b = 8$. Poiché $gcd(8, 65) = 1$ e $8^2 \equiv_{65} -1$, allora 65 è uno pseudoprimo forte rispetto alla base $b = 8$.
- Se invece consideriamo $b = 14$, osserviamo che, poiché $14^2 \equiv_{65} 1$, allora $14^{2^r} \equiv_{65} 1$ per $1 \leq r < 6$. Pertanto 65 non è uno pseudoprimo forte rispetto alla base $b = 14$.

Lemma 8.30. *Sia $n = 2^s t + 1$, t dispari, $s \geq 1$, uno pseudoprimo forte rispetto alla base b. Sia $p = 2^{s'} t' + 1$, t' dispari, $s' \geq 1$, un divisore primo di n. Se $b^{\frac{n-1}{2}} \equiv_n -1$, allora vale che:*

(1) $s' \geq s$;

(2) $\left(\frac{b}{p}\right) = \begin{cases} -1 \text{ se } s = s' \\ 1 \text{ se } s < s' \end{cases}$

Dimostrazione.

(1) Poiché $b^{\frac{n-1}{2}} = b^{2^{s-1}t} \equiv_n -1$ e t' è dispari, si ha $(b^{2^{s-1}t})^{t'} \equiv_n -1$. Pertanto $(b^{2^{s-1}t'})^t \equiv_n -1$. Siccome $p \mid n$, risulta

$$(b^{2^{s-1}t'})^t \equiv_p -1. \tag{8.7}$$

Supponiamo che $s' < s$. Allora

$$\left(b^{2^{s-1}t'}\right)^t = \left(b^{2^{s+s'-s'-1}t'}\right)^t = \left(b^{2^{s'-1}t'}\right)^{2^{s-s'}t}. \tag{8.8}$$

Poiché $b^{2^{s'-1}t'} = b^{\frac{p-1}{2}}$ e $b^{\frac{p-1}{2}} \equiv_p \left(\frac{b}{p}\right)$ per il **Teorema 8.7**, allora $b^{2^{s'-1}t'} \equiv_p \pm 1$. Ne segue che $\left(b^{2^{s'-1}t'}\right)^{2^{s-s'}t} \equiv_p 1$. Quindi, per (8.8), risulta $\left(b^{2^{s-1}t'}\right)^t \equiv_p 1$ ma ciò contraddice (8.7). Pertanto si ha $s' \geq s$.

(2) Se $s' = s$, segue che $b^{2^{s-1}t'} \equiv_p \left(\frac{b}{p}\right)$ per il **Teorema 8.7**. Allora $\left(b^{2^{s-1}t'}\right)^t \equiv_p \left(\frac{b}{p}\right)^t$ e quindi, da (8.7) risulta $\left(\frac{b}{p}\right)^t \equiv_p -1$ da cui segue, in modo banale, che $\left(\frac{b}{p}\right) = -1$, essendo t dispari. Ora supponiamo che $s' > s$. Poiché

$$\left(b^{2^{s'-1}t'}\right)^t = \left(b^{2^{s'+s-s-1}t'}\right)^t = \left(\left(b^{2^{s-1}t'}\right)^t\right)^{2^{s'-s}},$$

da (8.7) segue che $\left(b^{2^{s'-1}t'}\right)^t \equiv_p 1$ cioè $\left(b^{\frac{p-1}{2}}\right)^t \equiv_p 1$. Ora, dal **Teorema 8.7** si ha che $\left(\frac{b}{p}\right)^t \equiv_p 1$ e quindi $\left(\frac{b}{p}\right) = 1$, essendo t dispari.

□

Procedendo in modo analogo al **Lemma 8.30**, si prova il seguente Lemma:

Lemma 8.31. *Sia $n = 2^s t + 1$, t dispari, $s \geq 1$, uno pseudoprimo forte rispetto alla base b. Sia $p = 2^{s'} t' + 1$, t' dispari, $s' \geq 1$, un divisore primo di n. Se esiste $0 \leq r < s - 1$ tale che $b^{2^r t} \equiv_n -1$, allora vale che:*

(1) $s' \geq r$;

(2) $\left(\frac{b}{p}\right) = \begin{cases} -1 & \text{se } r = s' \\ 1 & \text{se } r < s' \end{cases}$

Proposizione 8.32. Se n è uno pseudoprimo forte rispetto alla base b, allora n è uno pseudoprimo di Eulero rispetto a b.

Dimostrazione. Sia $n = 2^s t + 1$, t dispari, $s \geq 1$, uno pseudoprimo forte rispetto alla base b. Analizziamo le tre seguenti (esaustive) possibiltà:

(1) $b^t \equiv_n 1$;
(2) $b^{2^{s-1}t} \equiv_n -1$ (i.e. $r = s - 1$);
(3) Esiste $0 \leq r < s - 1$ tale che $b^{2^r t} \equiv_n -1$.

(1) Supponiamo che valga (1), allora

$$b^{\frac{n-1}{2}} \equiv_n 1$$

poiché t divide $\frac{n-1}{2}$. D'altra parte,

$$\left(\frac{b}{n}\right)^t = \left(\frac{b^t}{n}\right) = 1$$

per la **Proposizione 8.17 (1)-(2)**. Pertanto, essendo t dispari, $\left(\frac{b}{n}\right) = 1$, e quindi

$$b^{\frac{n-1}{2}} \equiv_n \left(\frac{b}{n}\right),$$

cioè n è uno pseudoprimo di Eulero rispetto a b. Quindi, nel caso (1), la tesi è provata.

(2) Ora supponiamo che valga (2). Allora

$$b^{\frac{n-1}{2}} \equiv_n -1.$$

Sia $n = \prod_{p|n} p$. Per il **Lemma 8.30 (1)**, vale che $p = 2^{s'}t' + 1$, t' dispari, $s' \geq s \geq 1$. Per il **Lemma 8.30 (2)**, si ha che

$$\left(\frac{b}{n}\right) = \prod_{p|n}\left(\frac{b}{p}\right) = (-1)^k,$$

dove k è il numero, con ripetizione, dei divisori primi di n per cui valga $s' = s$. Essendo $b^{\frac{n-1}{2}} \equiv_n -1$, per provare l'asserto in questo caso è sufficiente provare che $\left(\frac{b}{n}\right) = -1$, cioè che k è dispari. Se $p = 2^{s'}t'+1$, t' dispari, $s' \geq 1$, è un generico primo divisore di n, si ha che:

- Se $s' > s$, allora $p \equiv_{2^{s+1}} 1$;
- Se $s' = s$, allora

$$p = 2^s t' + 1 = 2^s(2h' + 1) + 1 = 2^{s+1}h' + 2^s + 1$$

e quindi $p \equiv_{2^{s+1}} 2^s + 1$.

Ora $n = 2^s t + 1 = 2^s(2h + 1) + 1 = 2^{s+1}h + 2^s + 1$, e quindi

$$n \equiv_{2^{s+1}} 2^s + 1. \tag{8.9}$$

D'altra parte, essendo $n = \prod_{p|n} p$ si ha che

$$n \equiv_{2^{s+1}} (2^s + 1)^k. \tag{8.10}$$

Da (8.9) e (8.10) segue che

$$(2^s + 1)^k \equiv_{2^{s+1}} 2^s + 1. \tag{8.11}$$

Poiché

$$(2^s + 1)^k = \sum_{i=0}^{k}\binom{k}{i}2^{si} \equiv_{2^{s+1}} \binom{k}{0} + \binom{k}{1}2^s,$$

segue che

$$(2^s + 1)^k \equiv_{2^{s+1}} 1 + k2^s.$$

Allora per (8.11) vale

$$1 + 2^s k \equiv_{2^{s+1}} 1 + 2^s.$$

Quindi 2^{s+1} divide $2^s(k-1)$. Pertanto k è dispari, e quindi, anche nel caso (2) vale la tesi.

(3) Infine supponiamo che valga (3), allora

$$b^{\frac{n-1}{2}} \equiv_n 1.$$

Sia $n = \prod_{p|n} p$. Allora $n \equiv_{2^{r+1}} 1$. Per il **Lemma 8.31 (1)**, vale che se $p = 2^{s'}t' + 1$, t' dispari, $s' \geq 1$, è un generico primo divisore di n, allora risulta $s' \geq r$. Inoltre, dal **Lemma 8.31 (2)** segue che

$$\left(\frac{b}{n}\right) = \prod_{p|n}\left(\frac{b}{p}\right) = (-1)^\theta,$$

dove θ è il numero, con ripetizione, dei divisori primi di n per cui valga $s' = r$. Essendo $b^{\frac{n-1}{2}} \equiv_n 1$, per provare l'asserto in questo caso, è sufficiente provare che $\left(\frac{b}{n}\right) = 1$, cioè basta provare che θ è pari. Ragionando in modo analogo al caso precedente si ottiene $p \equiv_{2^{r+1}} 1$ o $p \equiv_{2^{r+1}} 1+2^r$ a seconda che il corrispondente s' sia maggiore o uguale ad r, rispettivamente. Quindi $n = \prod_{p|n} p$ è congruente sia a 1 che a $(1+2^r)^\theta$ modulo 2^{r+1} e pertanto $(1+2^r)^\theta \equiv_{2^{r+1}} 1$. Siccome $(1+2^r)^\theta \equiv_{2^{r+1}} 1 + 2^r\theta$, allora $1 + 2^r\theta \equiv_{2^{r+1}} 1$ e quindi θ è pari. Pertanto, l'asserto è dimostrato.

□

Remark 8.33. In generale non vale il viceversa della Proposizione precedente, come si può notare dal seguente esempio.

Esempio 8.34. Consideriamo $n = 561 = 3 \cdot 11 \cdot 17$ e $b = 2$. Risulta che n è uno pseudoprimo di Eulero ma non uno pseudoprimo forte rispetto alla base $b = 2$.

- Vale che $2^2 \equiv_3 1$ e per il **Piccolo Teorema di Fermat** $2^{10} \equiv_{11} 1$. Inoltre, siccome $17 = 2^4 + 1$, si ha che $2^8 \equiv_{17} 1$. Pertanto $2^{40} \equiv_{561} 1$ e quindi $2^{280} \equiv_{561} 1$, dove $280 = \frac{n-1}{2}$. Per la **Proposizione 8.18** vale che $\left(\frac{2}{561}\right) = 1$ e quindi $2^{280} \equiv_{561} \left(\frac{2}{561}\right)$. Pertanto 561 è uno pseudoprimo di Eulero rispetto alla base $b = 2$.
- Vale che $561 = 2^4 \cdot 35 + 1$. Se $2^{35} \equiv_{561} 1$ allora $2^5 \equiv_{561} 1$, poiché $2^{40} \equiv_{561} 1$, ma ciò è impossibile. Se $2^{70} \equiv_{561} -1$, allora $2^{30} \equiv_{561} -1$ e quindi $2^{10} \equiv_{561} -1$. Ciò è assurdo perchè 3 non divide $2^{10} + 1$. Allo stesso modo si prova che $2^{140}, 2^{280} \not\equiv_{561} -1$. Pertanto 561 non è uno pseudoprimo forte rispetto alla base $b = 2$.

□

Tuttavia, se $n \equiv_4 3$, vale anche il viceversa della **Proposizione 8.32** come risulta dalla seguente Proposizione:

Proposizione 8.35. Se $n \equiv_4 3$, allora n è uno pseudoprimo forte rispetto alla base b se e solo se n è uno pseudoprimo di Eulero rispetto a b.

Dimostrazione. Per la **Proposizione 8.32** è sufficiente provare che se n è uno pseudoprimo di Eulero rispetto alla base b, allora n è uno pseudoprimo forte rispetto alla base b. Siccome $n \equiv_4 3$, allora $n = 2t + 1$ con t dispari e quindi $b^t \equiv_n \left(\frac{b}{n}\right)$, i.e. $b^t \equiv_n \pm 1$.

□

Lemma 8.36. *Sia $(G,\cdot)$ un gruppo ciclico di ordine m, allora il numero degli elementi x in G tali che $x^k = 1$ è $d = gcd(m,k)$.*

Dimostrazione. Sia x in G tale che $x^k = 1$. Allora esiste un unico $0 \le j \le m-1$ tale che $x = g^j$, dove g è un fissato generatore di G. Pertanto $g^{jk} = 1$ e quindi $m \mid jk$. Segue che $\frac{m}{d} \mid j$, dove $d = gcd(m,k)$. Allora il numero delle soluzioni di $x^k = 1$ è uguale al numero dei multipli di $\frac{m}{d}$ minori o uguali a m, che è appunto d.

□

Lemma 8.37. *Siano $n = 2^s t + 1$, t dispari, un intero dispari e $p = 2^{s'} t' + 1$, t' dispari, un divisore primo di n. Allora il numero delle soluzioni dell'equazione $x^{2^r t} = -1$ in $GF(p)$ è 0 oppure $2^r(t,t')$ a seconda che $r \ge s'$ oppure $r < s'$, rispettivamente.*

Dimostrazione. Il numero delle soluzioni di $x^{2^r t} = -1$ è uguale al numero delle soluzioni di $x^{2^{r+1} t} = 1$ che non siano anche soluzioni di $x^{2^r t} = 1$.
Per il **Lemma 8.36**, il numero delle soluzioni è

$$(2^{r+1}t, 2^{s'}t') - (2^r t, 2^{s'}t') = (t', t)(2^{\max(r+1,s')} - 2^{\max(r,s')})$$

che è quindi 0 oppure $2^r(t,t')$ a seconda che sia $r \ge s'$ oppure $r < s'$, rispettivamente.

□

Teorema 8.38. *Sia n un intero composto dispari e sia Y l'insieme delle basi rispetto alle quali n è uno pseudoprimo forte. Allora*

$$|\mathcal{U}(\mathbb{Z}_n) - Y| \ge \frac{3}{4}\varphi(n).$$

Dimostrazione. Sia χ il rapporto tra il numero delle basi rispetto alle quali n è uno pseudoprimo forte e il numero totale delle basi. Per provare la tesi è sufficiente provare che $\chi \le \frac{1}{4}$. Si distinguono i seguenti casi:

(1) Esiste un primo p tale che $p^2 \mid n$;
(2) $n = pq$, con p, q primi distinti;
(3) $n = p_1 \cdots p_k$, $k > 2$, con $p_1, ..., p_k$ primi a due a due distinti.

Supponiamo che si verifichi il caso (1).
Allora n non è di Carmichael per la **Proposizione 8.24 (1)**. Sia quindi b una base tale che $b^{n-1} \equiv_n 1$. Allora $b^{n-1} \equiv_{p^2} 1$. Poiché $\mathcal{U}\left(\mathbb{Z}_{p^2}\right)$ è ciclico di ordine $p(p-1)$, allora il numero delle basi comprese tra 0 e p^2 tali che $b^{n-1} \equiv_{p^2} 1$ è $(p(p-1), n-1)$.

Poiché p non divide $n-1$, allora
$$(p(p-1), n-1) = (p-1, n-1) \leq p-1.$$
Procedendo in modo analogo, si ha che il numero delle basi comprese tra $p^2(h-1)$ e p^2h per $h = 1, \ldots, n/p^2$ con h fissato, è minore o uguale di $p-1$. Pertanto il numero delle basi b comprese tra 0 ed n tali che $b^{n-1} \equiv_{p^2} 1$ è minore o uguale a $\frac{n}{p^2}(p-1)$. Inoltre, il numero degli interi compresi tra 0 ed n che non sono divisibili per p^2 è $n - \frac{n}{p^2}$, i.e. $\frac{n}{p^2}(p^2-1)$. Sia X_1 l'insieme delle basi b rispetto alle quali n è uno pseudoprimo forte. Osserviamo che $|X_1|$ può essere maggiorata dal numero delle basi rispetto alle quali n è uno pseudoprimo. Tale numero, a sua volta, è minore o uguale del numero delle basi b tali che $b^{n-1} \equiv_{p^2} 1$.
Chiaramente $|X_1| \leq \frac{n}{p^2}(p-1)$. D'altra parte, $|X_2|$ cioè il numero totale delle basi b è minore o uguale al numero totale degli interi b compresi tra 0 ed n che non siano divisibili per p^2. Quindi $|X_2| \leq \frac{n}{p^2}(p-1)$. Pertanto segue che:
$$\chi = \frac{|X_1|}{|X_2|} \leq \frac{\frac{n}{p^2}(p-1)}{\frac{n}{p^2}(p^2-1)} = \frac{1}{p+1} \leq \frac{1}{4},$$
essendo p un primo che divide un intero composto dispari. Quindi vale la tesi.

Supponiamo che si verifichi il caso (2).
Quindi, $n = pq$, con p, q primi distinti, dove $p = 2^{s'}t' + 1$ e $q = 2^{s''}t'' + 1$, con $s', s'' \geq 1$ e t', t'' dispari. Senza perdere di generalità, possiamo supporre che $s'' \geq s'$. Sia $n = 2^s t + 1$, t dispari, $s \geq 1$, uno pseudoprimo forte rispetto ad una base b. Dalla **Definizione 8.29** segue che o $b^t \equiv_n 1$ oppure esiste $0 \leq r < s$ tale che $b^{2^r t} \equiv_n -1$.
Supponiamo che $b^t \equiv_n 1$. Allora $b^t \equiv_p 1$ e $b^t \equiv_q 1$. Dal **Lemma 8.36**, segue che il numero dei $0 < b < p$ e dei $0 < b < q$ per cui si verifica questa possibilità è (t', t) e (t'', t), rispettivamente. Quindi il numero dei $0 < b < n$ per cui valga $b^t \equiv_n 1$ è $(t', t)(t'', t) \leq t't''$.
Se invece esiste $0 \leq r < s$ tale che $b^{2^r t} \equiv_n -1$, allora $b^{2^r t} \equiv_p -1$ e $b^{2^r t} \equiv_q -1$. Sia $n_{p,r}$ il numero delle soluzioni comprese tra 0 e p dell'equazione congruenziale $b^{2^r t} \equiv_p -1$. Allora, dal **Lemma 8.37**, segue che $n_{p,r} = 2^r(t, t')$ oppure $n_{p,r} = 0$ a seconda che $r < s'$ o $r \geq s'$, rispettivamente ($s' = \min(s', s'')$). Definendo in modo analogo $n_{q,r}$, si ha che $n_{q,r} = 2^r(t, t'')$ oppure $n_{q,r} = 0$ a seconda che $r < s'$ o $r \geq s'$, rispettivamente. Quindi il numero dei b tali che $b^{2^r t} \equiv_n -1$ è $n_{p,r} \cdot n_{q,r} \leq 2^r(t, t')2^r(t, t'') = 4^r t't''$. Siccome $\varphi(n) = (p-1)(q-1) = 2^{s'+s''}t't''$, allora vale che
$$\begin{aligned} \chi &\leq \frac{t't'' + \sum_{r=0}^{s-1} n_{p,r}n_{q,r}}{2^{s'+s''}t't''} = \frac{t't'' + \sum_{r=0}^{s'-1} n_{p,r}n_{q,r}}{2^{s'+s''}t't''} \leq \frac{t't'' + \sum_{r=0}^{s'-1} 4^r t't''}{2^{s'+s''}t't''} \\ &= \frac{1}{2^{s'+s''}}\left(1 + \sum_{r=0}^{s'-1} 4^r\right) = \frac{1}{2^{s'+s''}}\left(1 + \frac{4^{s'}-1}{4-1}\right) \end{aligned} \tag{8.12}$$
Se $s'' > s'$, allora (1) implica
$$\chi < \frac{1}{2^{2s'+1}}\left(\frac{3 + 2^{2s'} - 1}{3}\right) = \frac{1}{2^{2s'}3} + \frac{1}{6} \leq \frac{1}{12} + \frac{1}{6} = \frac{1}{4}.$$
Quindi, in questo sottocaso, vale l'asserto.

Consideriamo il caso in cui $s'' = s'$. Proviamo che $(t,t') < t'$ oppure che $(t,t'') < t''$. Supponiamo per assurdo che $(t,t') = t'$ e $(t,t'') = t''$. Allora $t' \mid t$ e $t'' \mid t$. Dal fatto che t' divide t, segue che $t' \mid n-1$ e quindi $t' \mid q-1$, essendo $n-1 = p(q-1)+p-1$ e t' un divisore di $p-1$. Allora $t' \mid t''$, siccome $q-1 = 2^{s''}t''$. Analogamente, si prova che $t'' \mid t'$ e quindi si ha che $t' = t''$. Pertanto, essendo $s' = s''$, segue che $p = q$ ma ciò è assurdo.
Pertanto si ha che $(t,t') \leq \frac{1}{3}t'$ oppure $(t,t'') \leq \frac{1}{3}t''$. Allora $(t,t')(t,t'') \leq \frac{1}{3}tt'$ e quindi segue che:

$$\begin{aligned}\chi &\leq \frac{\frac{1}{3}t't'' + \sum_{r=0}^{s-1} 4^r (t,t')(t,t'')}{2^{2s'}t't''} = \frac{\frac{1}{3}t't'' + \frac{1}{3}\sum_{r=0}^{s'-1} 4^r t't''}{2^{2s'}t't''} \\ &= \frac{\frac{1}{3} + \frac{1}{3}\sum_{r=0}^{s'-1} 4^r}{2^{2s'}} = \frac{1}{2^{2s'}3}\left(1 + \frac{2^{2s'}-1}{2^2-1}\right) = \frac{1}{2^{2s'}3^2}\left(2^{2s'}+2\right) \\ &= \frac{1}{9} + \frac{1}{18} < \frac{1}{4}.\end{aligned}$$

Ciò completa la dimostrazione nel caso (2).

Supponiamo che si verifichi il caso (3).
Quindi, $n = p_1 \cdot \ldots \cdot p_k$, $k > 2$, con $p_1, ..., p_k$ primi a due a due distinti. Chiaramente $p_i = 2^{s_i}t_i + 1$, con $s_i, t_i \geq 1$, t_i dispari. Possiamo assumere senza perdere di generalità che $s_1 = \min(s_1, ..., s_k)$. Sia $n = 2^s t + 1$, t dispari, $s \geq 1$, uno pseudoprimo forte rispetto ad una base b. Dalla **Definizione 8.29** segue che o $b^t \equiv_n 1$ oppure esiste $0 \leq r < s$ tale che $b^{2^r t} \equiv_n -1$.
Supponiamo che $b^t \equiv_n 1$, allora $b^t \equiv_{p_i} 1$ per ogni $i = 1, \ldots, k$. Dalla **Proposizione 8.2 (2)**, segue che il numero dei b compresi tra 0 ed n per cui $b^t \equiv_n 1$ è uguale al $\prod_{i=1}^k (t_i, t) \leq \prod_{i=1}^k t_i$.
Se invece esiste un $0 \leq r < s$ per cui $b^{2^r t} \equiv_n -1$, allora $b^{2^r t} \equiv_{p_i} -1$ per ogni i. Sia $n_{p_i,r}$ il numero delle soluzioni comprese tra 0 e p_i dell'equazione congruenziale $b^{2^r t} \equiv_{p_i} -1$. Allora, dal **Lemma 8.37**, segue che $n_{p_i,r} = 2^r(t_i, t)$ oppure $n_{p_i,r} = 0$ a seconda che $r < s_1$ o $r \geq s_1$, rispettivamente.
Quindi il numero dei b tali che $b^{2^r t} \equiv_n -1$ è $\prod_{i=1}^k n_{p_i,r} \leq \prod_{i=1}^k 2^r (t_i, t) \leq 2^{rk} \prod_{i=1}^k t_i$.
Siccome

$$\varphi(n) = \prod_{i=1}^k \varphi(p_i) = \prod_{i=1}^k (p_i - 1) = \prod_{i=1}^k 2^{s_i} t_i = 2^{s_1+\ldots+s_k} \prod_{i=1}^k t_i,$$

quindi

$$\begin{aligned}\chi &\leq \frac{\prod_{i=1}^k t_i + \prod_{r=0}^{s-1} 2^{rk} \prod_{i=1}^k t_i}{2^{s_1+\ldots+s_k} \prod_{i=1}^k t_i} = \frac{\prod_{i=1}^k t_i + \prod_{r=0}^{s_1-1} 2^{rk} \prod_{i=1}^k t_i}{2^{s_1+\ldots+s_k} \prod_{i=1}^k t_i} \\ &= \frac{1}{2^{s_1+\cdots+s_k}}\left(1 + \sum_{r=0}^{s_1-1} 2^{kr}\right) \leq \frac{1}{2^{ks_1}}\left(1 + \frac{2^{ks_1}-1}{2^k-1}\right) \\ &= \frac{1}{2^{ks_1}}\left(\frac{2^k-2}{2^k-1} + \frac{2^{ks_1}}{2^k-1}\right) = \frac{1}{2^{ks_1}} \cdot \frac{2^k-2}{2^k-1} + \frac{1}{2^k-1} \\ &\leq \frac{1}{2^k} \cdot \frac{2^k-2}{2^k-1} + \frac{1}{2^k-1} = 2^{1-k} \leq \frac{1}{4},\end{aligned}$$

essendo $k > 2$. Ciò completa la dimostrazione.

□

Remark 8.39. La probabilità che un intero composto dispari n sia uno pseudoprimo forte rispetto a k basi distinte è al più $\frac{1}{4^k}$.

8.2 Test di Primalità

In questa sezione vederemo i Testi di primalità di **Solovay-Strassen** e **Miller-Rabin** che si basano sulle definizioni di Pseudoprimi di Eulero e Pseudoprimi forti, rispettivamente.

Figura 8.1: Robert Martin Solovay (1938), Volker Strassen (1936), Gary Miller e Michael Oser Rabin (1931)

Gli algoritmi hanno le seguenti caratteristiche

- **Vantaggio:** Questi algoritmi sono computazionalmente efficienti, ovvero un intero n è testato in un tempo $O(\log^3 n)$.
- **Svantaggio:** Gli algoritmi forniscono come output che n è un numero primo, quando in realtà non lo è. Tuttavia, la probabilià di commettere un errore può essere ridotta sotto una fissata soglia eseguendo l'algoritmo un certo numero di volte.

Una domanda importante è quanti interi casuali bisogna testare prima di trovarne uno che sia primo.

Siano N un intero positivo e $\pi(N) = |\{p \leq N : p \text{ primo}\}|$. Il **Teorema dei numeri Primi** asserisce che $\pi(N) \simeq N/\ln N$. Quindi la probabilià che un intero $p \leq N$ sia primo è $1/\ln N$. Se p è di circa 512 bit, allora la suddetta probabilità è $1/\ln 2^{512} \simeq 1/355$. Pertanto, in media, si trova un numero primo ogni 355 interi casuali.

In realtà, la probabiltà è raddoppiata poichè tali primi sono numeri dispari. Quindi la probabilià è di circa 2/355. Pertanto, la genesi dei numeri primi per la costruzione dell'RSA è fattibile.

Definizione 8.40. **(Problema Decisionale)**
Un **problema decisionale** è una domanda a cui deve essere risposto con un 'sì' o un 'no'.

Definizione 8.41. **(Algoritmo Montecarlo)**
Un **Algoritmo Montecarlo Orientato verso il si con probabilità di errore** ε è un algoritmo randomizzato utilizzato per un problema decisionale in cui la risposta 'si' è corretta, mentre la risposta 'no' può essere non corretta con probabilità al più ε.

Si consideri il seguente problema decisionale

Problema 8.42. **(Interi Composti)**

Istanza: Un intero positivo $n \geq 2$
Domanda: n è un intero composto?

Vediamo ora il Test di Primalià di **Soloway-Strassen**:

Algoritmo 8.43. **(Soloway-Strassen (n))**

Si scelga un intero a compreso tra 1 e $n-1$
$x \longleftarrow \left(\frac{a}{n}\right)$
if $x = 0$
then return ("n è un intero composto")
$y \longleftarrow a^{(n-1)/2} \ mod \ n$
if $x \equiv y \ mod \ n$
then return ("n è primo")
else return ("n è un intero composto")

Teorema 8.44. *Il Test di Primalità di Soloway-Strassen è un algoritmo Montecarlo orientato verso il si con probabilità di errore al più* $1/2$.

Dimostrazione. Dall'analisi dell'**Algoritmo 8.43** segue immediatamente che l'output "n è un intero composto" è sempre corretta, mentre segue dal **Teorema 8.26** che la probabilità che l'output "n è primo" sia scorretto è minore o uguale a $1/2$.

□

Teorema 8.45. *Il Test di Primalità di Soloway-Strassen ha complessità* $O(\log^3 n)$.

Dimostrazione. Il calcolo del simbolo di Jacobi consiste di riduzioni modulari e la fattorizzazione rispetto alle potenze di 2. Siccome ogni intero è rappresentato in binario, la la fattorizzazione rispetto alle potenze di 2 consiste nel determinare il numero degli 0 finali. Quindi il calcolo del simbolo di Jacobi si riduce al numero di riduzioni modulari. Siccome ogni riduzione modulare ha cmplessità $O(\log^2 n)$ e il numero delle riduzioni è al più pari al numero delle cifre di n, vale che il simbolo di Jacobi ha complessità $O(\log^3 n)$. D'altra parte, $a^{(n-1)/2} \ mod \ n$ ha complessità $O(\log^3 n)$. Pertanto, il Test di Primalità di Soloway-Strassen ha complessità $O(\log^3 n)$.

□

Remark 8.46. Un'analisi più precisa mostra che in realtà il Test di Primalità di Soloway-Strassen ha complessità $O(\log^2 n)$.

Supponiamo di aver generato un intero causale dispari n e vogliamo testarne la sua primalità attraverso l'**Algoritmo di Soloway-Strassen**. Dopo averlo testato m volte, quale è la confidenza che n sia primo? Saremmo tentati, sulla base del **Teorema 8.26** che avendolo superato m volte la probabilià che n sia primo è $1 - 1/2^m$.

Nella realtà deve essere fatta una più precisa analisi come segue. Si considerino i seguenti eventi

E_1 : un intero causale dispari n di fissata grandezza è composto.

E_2 : l'**Algoritmo 8.43** fornisce in output "n è primo" m volte in successione.

Dal **Teorema 8.26** segue che $\boldsymbol{P}[E_2/E_1] \leq 1/2^m$. Noi invece siamo interessati a $\boldsymbol{P}[E_1/E_2]$. Dal **Teorema di Bayes** segue che

$$\boldsymbol{P}[E_1/E_2] = \frac{\boldsymbol{P}[E_2/E_1]\boldsymbol{P}[E_1]}{\boldsymbol{P}[E_2]} = \frac{\boldsymbol{P}[E_2/E_1]\boldsymbol{P}[E_1]}{\boldsymbol{P}[E_2/E_1]\boldsymbol{P}[E_1] + \boldsymbol{P}[E_2/E_1^c]\boldsymbol{P}[E_1^c]}.$$

Dal **Teorema dei Numeri Primi** segue che se $N \leq n \leq 2N$, il numero dei primi (dispari) compresi tra N e $2N$ è approssimativamente a

$$\frac{2N}{\ln 2N} - \frac{N}{\ln N} \simeq \frac{N}{\ln N} \simeq \frac{n}{\ln n}.$$

Siccome il numero degli interi dispari tra N è $2N$ è $N/2 \simeq n/2$, allora

$$\boldsymbol{P}[E_1^c] \simeq \frac{n}{\ln n} / \frac{n}{2} = \frac{2}{\ln n}.$$

Inoltre, dal **Teorema di Eulero (Teorema 8.7)** segue che $\boldsymbol{P}[E_2/E_1^c] = 1$.

Quindi

$$\begin{aligned}
\boldsymbol{P}[E_1/E_2] &= \frac{\boldsymbol{P}[E_2/E_1]\left(1-\frac{2}{\ln n}\right)}{\boldsymbol{P}[E_2/E_1]\left(1-\frac{2}{\ln n}\right)+\frac{2}{\ln n}} \\
&= \frac{\boldsymbol{P}[E_2/E_1](\ln n-2)}{\boldsymbol{P}[E_2/E_1](\ln n-2)+2} \\
&\leq \frac{2^{-m}(\ln n-2)}{2^{-m}(\ln n-2)+2} \\
&= \frac{\ln n-2}{(\ln n-2)+2^{m+1}}.
\end{aligned}$$

Nella seguente tabella sono tabulate le funzioni 2^{-m} e $\frac{\ln n-2}{(\ln n-2)+2^{m+1}}$.

m	2^{-m}	bound sulla probabilità di errore
1	0.500	0.989
2	0.200	0.978
5	0.312×10^{-1}	0.847
10	0.977×10^{-3}	0.147
20	0.954×10^{-6}	0.168×10^{-3}
30	0.931×10^{-9}	0.164×10^{-6}
50	0.888×10^{-15}	0.157×10^{-12}
100	0.789×10^{-30}	0.139×10^{-27}

I primi utilizzati per la costruzione del crittosistema RSA devono essere di circa 512 cifre binarie. Sia n un intero candidato ad essere uno dei due primi per la costruzione del crittosistema RSA. Quindi sia $n \simeq 2^{512} \simeq e^{355}$ un intero da testare, allora

$$\frac{\ln n-2}{(\ln n-2)+2^{m+1}} \simeq \frac{353}{353+2^{m+1}}$$

per $50 \leq m \leq 100$ la probabilità di commettere un errore è veramente piccola.

Algoritmo 8.47. **(Miller-Rabin (n))**
Si scriva n come $2^k m$ con m dispari
Si scelga un intero casuale a compreso tra 1 e $n-1$
$b \longleftarrow a^m \ mod\ n$
if $b \equiv 1 \ mod\ n$
then return ("n è primo")
for $i \longleftarrow 0$ **to** $k-1$
do { **if** $b \equiv -1 \ mod\ n$ **then return** ("n è primo") **else** $b \longleftarrow b^2 \ mod\ n$
return ("n è composto")

Teorema 8.48. *Il Test di Primalità di Miller-Rabin è un algoritmo Montecarlo orientato verso il si con probabilità di errore al più* $1/4$ *e complessità* $O(\log^3 n)$.

Dimostrazione. Dall'analisi dell'**Algoritmo 8.47** segue immediatamente che l'output "n è un intero composto" è sempre corretto, mentre segue dal **Teorema 8.38** che la probabilità che l'output "n è primo" sia scorretto è minore o uguale a $1/4$. Infine, è facile vedere che la complessità è $O(\log^3 n)$.

□

Procedendo in modo analogo all'algoritmo di **Soloway-Strassen** vale

E_1 : un intero causale dispari n di fissata grandezza è composto.

E_2 : l'**Algoritmo 8.47** fornisce in output "n è primo" m volte in successione.

Segue che

$$\begin{aligned} \boldsymbol{P}[E_2/E_1] &\leq 1/2^{2m}. \\ \boldsymbol{P}[E_1/E_2] &\leq \frac{\ln n - 2}{(\ln n - 2) + 2^{2m+1}}. \end{aligned}$$

9 Metodi di Attacco al Crittosistema RSA

9.1 Radici quadrate modulo un intero

I principali metodi di attacco al crittosistema RSA sono quelli che si basano sulla fattorizzazione di n. Prima di vedere questo, vediamo alcune informazioni di aritmetica modulare che saranno utili per il seguito.

Siano a, n due interi, n dispari tale che $\gcd(n, a) = 1$. Vogliamo determinare il numero N delle soluzioni $y \in \mathbb{Z}_n$ tale che

$$y^2 \equiv a \ mod \ n. \tag{9.1}$$

Abbiamo già visto che, se n è primo allora N è 2 o 0 a seconda che il simbolo di Legendre $\left(\frac{a}{n}\right)$ sia uguale a 1 o -1, rispettivamente.

Proposizione 9.1. Se $n = p^e$, con p primo dispari tale che $\gcd(a, p) = 1$ ed e intero positivo, allora N è 2 o 0 a seconda che il simbolo di Legendre $\left(\frac{a}{p}\right)$ sia uguale a 1 o -1, rispettivamente.

Dimostrazione. Da (9.1) si ha $y^2 \equiv a \ mod \ p$, quindi $N = 0$ se $\left(\frac{a}{p}\right) = -1$. Pertanto, supponiamo che $\left(\frac{a}{p}\right) = 1$ e sia y_0 un intero tale che $y_0^2 \equiv a \ mod \ p$. Siccome $y_0^2 \equiv a \ mod \ p^{e-1}$, allora esiste $k \in \mathbb{Z}$ tale che $y_0^2 = a + kp^{e-1}$. Siccome $\gcd(2y_0, p) = 1$, allora sia $d \in \mathbb{Z}$ tale che $2y_0 d \equiv 1 \ mod \ p^{e-1}$ e sia quindi

$$y_1 = y_0 - kdp^{e-1}.$$

Quindi

$$\begin{aligned} y_1^2 &= \left(y_0 - kdp^{e-1}\right)^2 = \\ &= y_0^2 + k^2 d^2 p^{2(e-1)} - kp^{e-1} = \\ &= a + kp^{e-1} + k^2 d^2 p^{2(e-1)} - kp^{e-1} = \\ &= a + k^2 d^2 p^{2(e-1)} \end{aligned}$$

e quindi $y_1^2 \equiv a \ mod \ p^e$. Pertanto, $N = 2$.

□

Teorema 9.2. *Sia $n > 1$ un intero dispari avente fattorizzazione*

$$n = \prod_{i=1}^{\ell} p_i^{e_i},$$

dove p_i è primo e e_i è un intero positivo, $i = 1, ..., \ell$. Se $\gcd(n, a) = 1$, allora N è 2^ℓ se per ogni $i = 1, ..., \ell$ il simbolo di Legendre $\left(\frac{a}{p_i}\right) = 1$, e 0 altrimenti.

Dimostrazione. Se esiste $i_0 \in \{1, ..., \ell\}$ tale che $\left(\frac{a}{p_{i_0}}\right) = -1$, allora $N = 0$ siccome (9.1) implica $y^2 \equiv a \ mod \ p_{i_0}$. Pertanto, $\left(\frac{a}{p_i}\right) = 1$ per ogni $i = 1, ..., \ell$. Pertanto, $y^2 \equiv a \ mod \ p_i^{e_i}$ ammette 2 soluzioni b_{i1} e b_{i2} per la **Proposizione 9.1**. Quindi, Per il **Teorema Cinese dei Resti**, per ogni $(b_{1j_1}, ..., b_{\ell j_\ell})$ il sistema congruenziale

$$\begin{cases} y \equiv b_{1j_1} \ mod \ p_1^{e_1} \\ \vdots \\ y \equiv b_{\ell j_\ell} \ mod \ p_\ell^{e_\ell} \end{cases}$$

ha un unica soluzione modulo n. Pertanto, $N = 2^\ell$ siccome il numero di tali ℓ-ple $(b_{1j_1}, ..., b_{\ell j_\ell})$ è proprio 2^ℓ.

□

9.2 Metodi di attacco al crittosistema RSA

Il metodo più ovvio per violare il crittosistema RSA consiste nel fattorizzare. Vediamo alcuni metodi utilizzati per fattorizzare n.

Divisione: Siccome n è composto, esiste un primo p tale che $p \leq \lfloor \sqrt{n} \rfloor$. Pertanto, tale metodo consiste nel dividere n con ogni intero dispari minore o uguale a $\lfloor \sqrt{n} \rfloor$. Siffatto metodo è considerato ragionevole se $n < 10^{12}$.

Algoritmo $p-1$ di Pollard (1974): Siano n, B interi. Supponiamo che esista un primo p divisore di n tale che se $p - 1 = \prod_{i=1}^{k} q_i^{\alpha_i}$, allora $q_i^{\alpha_i} \leq B$. Pertanto $(p-1) \mid B!$. Quindi, esiste un $2 \leq j_0 \leq B!$ tale che $j_0 = k(p-1)$. Pertanto, per il **piccolo Teorema di Fermat**, vale che $2^{j_0} \equiv 1 \ mod \ p$. Quindi, $p \mid \gcd(2^{j_0} - 1, n)$. Se $\gcd(2^{j_0} - 1, n) < n$, allora $\gcd(2^{j_0} - 1, n)$ è un fattore proprio di n.

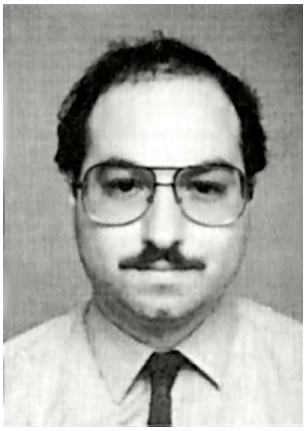


Figura 9.1: Jonathan Jay Pollard (1954)

Algoritmo 9.3. **($p-1$ di Pollard (n, B))**

$a \longleftarrow 2$
for $j \longleftarrow 2$ **to** B
do $\begin{cases} a \longleftarrow a^j \ mod\ n \\ d \longleftarrow \gcd(a^j \ mod\ n - 1, n) \end{cases}$
if $1 < d < n$
then return (d)
else return ("insuccesso")

Chiaramente, l'**Algoritmo 9.3** è di tipo Las Vegas. Infatti, perché esso funzioni, n deve ammettere un divisore primo p tale che $p-1$ sia costituito da potenze di primi piccoli, minori o uguali al bound B prefissato.

Esempio 9.4. Si consideri $n = 15770708441$ e $B = 180$ allora $a = 11620221425$ e $\gcd(a-1, n) = 135979$. Quindi

$$15770708441 = 135979 \times 115979$$

e l'algoritmo funziona siccome $135978 = 2 \times 3 \times 131 \times 173$. Quindi, in realtà il metodo funziona per ogni $B \geq 173$.

Nell'algoritmo $p-1$ si eseguono $B-1$ esponenziazioni modulari e $B-1$ calcoli di gcd. Ogni esponensiazione modulare richiede $\ln B \ln^2 n$ bit e ogni calcolo di gcd richiede $\ln^2 n$. Pertanto, la complessità dell'algoritmo $p-1$ è $O(B \ln B \ln^2 n)$. Chiaramente per B vicino a $\sqrt{n}$ la probabilità di successo aumenta ma la velocità di esecuzione e circa quella delle divisioni .
Si noti che, per rendere l'RSA immune da siffatti tipi di attacco si considera $n = pq$ con $p = 2p_1 + 1$ e $q = 2q_1 + 1$ con p_1, q_1 primi elevati.

- **Algoritmo di Lenstra.** È un generalizzazione del metodo di $p-1$ di Pollard basato sulle curve elittiche (lo vedremo più avanti).
- **Algoritmo Rho di Pollard.** L'idea che sta alla base dell'algoritmo è che se p il minimo divisore primo di n ed esistono due interi x, x' minori di n tali che $x \neq x'$ e $x \equiv x' \ mod\ p$, allora

$$p \leq \gcd(x - x', n) < n.$$

  E' così determinato un fattore non banale di n.

Più pecisamente, si sceglie $X \subseteq \mathbb{Z}_n$ e per ogni coppia (x, x') di elementi distinti di X si calcola $\gcd(x - x', n)$. L'algoritmo ha successo se l'applicazione $x \longmapsto x \ mod\ p$ ammette una collisione in X. Utilizzando il **paradosso del compleanno**, ciò si verifica con una probabilità de 50% se $|X| \simeq 1.17\sqrt{p}$. Tuttavia, siccome p è sconosciuto, la collisione è determinata solo valutando $\gcd(x-x', n)$ per ogni coppia (x, x') di elementi distinti di X.

Pertanto, per determinare una collisione, il numero dei gcd che bisogna calcolare è $\binom{|X|}{2} > p/2$ che è un numero elevato. Per ridurre sia il numero dei gcd da calcolare che la memoria da utilizzare, l'algoritmo Rho di Pollard procede come segue:

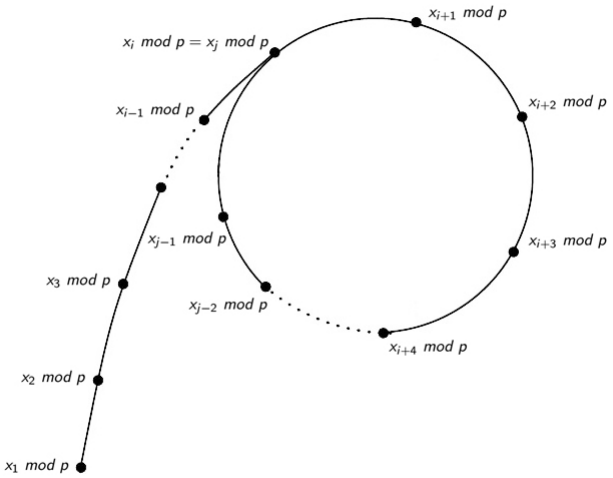
1. Si considera $f \in \mathbb{Z}[x]$ e $x_1 \in \mathbb{Z}_n$ (generalmente si considera $f(x) = x^2 + a$ e nella maggior parte dei casi $a = 1$);
2. Si considera la sequenza definita da $x_{j+1} \equiv f(x_j) \ mod\ n$ per $2 \leq j \leq m-1$ (si determina così $X = \{x_1, ..., x_m\}$ di m elementi che assumiamo essere tutti distinti);
3. Ogni volta che si determina un x_j si valuta $\gcd(x_j - x_i, n)$ per ogni $j < i$.

Lemma 9.5. *Vale che*

$$x_i \equiv x_j \ mod\ p \Rightarrow \forall \delta \in \mathbb{N} : x_{i+\delta} \equiv x_{j+\delta} \ mod\ p. \tag{9.2}$$

Dimostrazione. Siccome esiste una collisione $x_i \equiv x_j \ mod\ p$ per qualche $2 \leq i, j \leq m-1$ allora $f^\delta(x_i) \equiv f^\delta(x_j) \ mod\ p$, ovvero $x_{i+\delta} \equiv x_{j+\delta} \ mod\ p$, per ogni $\delta \geq 0$.

□



Vediamo un esempio.

Esempio 9.6. Sia $n = 7171 = 71 \times 101$ e $f(x) = x^2 + 1$ e $x_1 = 1$ allora la sequenza degli x_i è

1	2	5	26	667	6557	4105
6347	4903	2218	219	4936	4210	4560
4872	375	4377	4389	2016	5471	88

e ridotti modulo 71 si ha

1	2	5	26	28	25	58
28	4	17	6	37	21	16
44	20	46	58	28	4	17

e quindi la prima collisione è $x_7 \ mod\ 71 = x_{18} \ mod\ 71 = 58$.
Infatti, $\gcd(x_7 - x_{18}, 7171) = \gcd(4105 - 4389, 7171) = 71$.

Lemma 9.7. ***(Trucco di Floyd)***
Se $x_i \equiv x_j \ mod\ p$ allora esiste $i' \in \{i, i+1, ..., j-1\}$ tale che $x_{i'} \equiv x_{2i'} \ mod\ p$.

Dimostrazione. Sia $\ell = j - i$, poichè $\{i, i+1, \cdots, j-1\}$ è un sistema completo di residui modulo ℓ, esiste $i' \in \{i, i+1, \cdots, j-1\}$ tale che $i' = k_0\ell$. Sia $\delta = i' - i$, allora vale

$$x_{i'} \equiv x_{i'+\ell} \ mod\ p.$$

Ora applicando (9.2) con i' al posto di i, $i'+\ell$ al posto di j ed ℓ al posto di δ, si ha $x_{i'+\ell} \equiv x_{i'+2\ell} \ mod\ p$. Pertanto, $x_{i'} \equiv x_{i'+2\ell} \ mod\ p$ e induttivamente

$$x_{i'} \equiv x_{j'} \ mod\ p \text{ con } j' = i' + k\ell, k \in \mathbb{N}.$$

Se $k = k_0$, allora $j' = 2i'$ e quindi vale

$$x_{i'} \equiv x_{2i'} \ mod\ p.$$

□

Algoritmo 9.8. **(Rho di Pollard** (n, x_1)**)**

esternal f
$x \longleftarrow x_1$
$x' \longleftarrow f(x) \ mod\ n$
$p \longleftarrow \gcd(x - x', n)$
while $p = 1$
do
- **commenti:** nella i-esima iterazione
- $x \longleftarrow f(x) \ mod\ n$
- $x' \longleftarrow f(x') \ mod\ n$
- $x' \longleftarrow f(x') \ mod\ n$
- $p \longleftarrow \gcd(x - x', n)$

if $p = n$
then return ("insuccesso")
else return (p)

Sulla base del **Lemma 9.7** si cercano collisioni del tipo $x_{i'} \equiv x_{2i'} \ mod\ p$ per qualche $i' \in \{i, i+1, ..., j-1\}$. Quindi, all'iterazione i-esima si calcola solo $\gcd(x_{2i} - x_i, n)$. Essendo, il numero di iterazioni per determinare un fattore p primo di n e di circa $\sqrt{p}$. Si noti che in questo modo non si determina la prima collisione, se ne esistono, a vantaggio del minor numero di gcd da calcolare.

Ritornando all'esempio precedente, si ha che la prima collisione per $i = 7$ e $j = 18$ quindi $\ell = 11$, il primo $i' \geq i$ tale che si abbia una collisione è $i = 11$. Infatti $\gcd(x_{22} - x_{11}, n) = \gcd(7745 - 219, 7171) = 71$.

Si prova che la probabilià di successo è circa $p/n < 1/\sqrt{n}$ e la complessità dell'**Algoritmo Rho di Polard** è $O(\sqrt[4]{n} \ln n)$.

9.3 Algoritmo di Dixon per i quadrati casuali

L'idea che sta alla base è molto semplice: è quella di determinare due interi x, y tali che $x \not\equiv \pm y \ mod\ n$ ma $x^2 \equiv y^2 \ mod\ n$. Pertanto $n \mid (x+y)(x-y)$ ma non divide $x-y$ o $x+y$. Pertanto $\gcd(n, x-y)$ e $\gcd(n, x+y)$ sono fattori non banali di n.

L'algoritmo opera come segue:

(1) Si fissa $\mathcal{B} = \{p_1, ..., p_b\}$ un sottoinsieme (**base**) di primi, eventualmente unito a $\{-1\}$.

(2) Si determinano $z_1, ..., z_c$ interi casuali con $c > b$ e di ognuno di essi calcola il quadrato e lo si riduce modulo n. Generalmente, interi della forma $j + \lceil \sqrt{kn} \rceil$ con $j = 0, 1, 2...$ e $k = 1, 2, ..$ che tendono produrre corrispondenti $z^2 \ mod\ n$ relativamente piccoli e quindi completamente fattorizzabili rispetto i primi della base. Oppure interi oppure della forma $\lfloor \sqrt{kn} \rfloor$, tali che $z^2 \ mod\ n$ è vicino ad n e quindi $-z^2 \ mod\ n$ è potenzialmente completamente fattorizzabili rispetto i primi della base $\mathcal{B}$.

(3) Per ogni $1 \leq j \leq c$

$$z_j^2 \equiv p_1^{\alpha_{1j}} \times p_2^{\alpha_{2j}} \cdots \times p_b^{\alpha_{bj}} \ mod\ n$$

e si considera il vettore di $\mathbb{Z}_2^b$ definito da

$$\vec{a}_i = (\alpha_{1j} \ mod\ 2, \alpha_{2j} \ mod\ 2, \ldots, \alpha_{bj} \ mod\ 2)$$

(4) Poichè $c > b$, i vettori $\vec{a}_1, \ldots \vec{a}_c$ sono linearmente dipendenti. Quindi esiste $X \subseteq \{1, .., c\}$ tali $\sum_{j \in X} \vec{a}_j = \vec{0}$ e pertanto

$$\prod_{j \in X} z_j^2 \equiv \prod_{i=1}^{b} p_i^{\sum_{j \in X} \alpha_{ij}} \ mod\ n$$

siccome $\sum_{j \in X} \vec{a}_j = \vec{0}$, allora $\sum_{j \in X} \alpha_{ij} = 2k_i$ per ogni $j \in X$. Quindi, posto $x = \prod_{j \in X} z_j$ e $y = \prod_{i=1}^{b} p_i^{k_i}$ vale che $x^2 \equiv y^2 \ mod\ n$. Da qui si procede al calcolo di $\gcd(n, x-y)$ o di $\gcd(n, x+y)$.

Esempio 9.9. Fattorizziamo $n = 1829$ attraverso i quadrati casuali di Dixon.

Sia $\mathcal{B} = \{-1, 2, 3, 5, 7, 11, 13\}$ calcoliamo $\sqrt{n} = 42.77$, $\sqrt{2n} = 60.48$, $\sqrt{3n} = 74.03$ e $\sqrt{4n} = 85.53$. Consideriamo gli interi

$$z = 42, 43, 61, 62, 74, 75, 85, 86.$$

Allora

$$\begin{aligned} z_1^2 &\equiv 42^2 \equiv -65 \equiv -1 \times 5 \times 13 \ \textit{mod}\ 1829 \\ z_2^2 &\equiv 43^2 \equiv 20 \equiv 2^2 \times 5 \ \textit{mod}\ 1829 \\ z_3^2 &\equiv 61^2 \equiv 63 \equiv 3^2 \times 7 \ \textit{mod}\ 1829 \\ z_4^2 &\equiv 74^2 \equiv -11 \equiv -1 \times 11 \ \textit{mod}\ 1829 \\ z_5^2 &\equiv 85^2 \equiv -91 \equiv -1 \times 7 \times 13 \ \textit{mod}\ 1829 \\ z_6^2 &\equiv 86^2 \equiv 80 \equiv 2^4 \times 5 \ \textit{mod}\ 1829 \end{aligned}$$

Quindi

$$\begin{aligned} \vec{a}_1 &= (1,0,0,1,0,0,1) \\ \vec{a}_2 &= (0,0,0,1,0,0,0) \\ \vec{a}_3 &= (0,0,0,0,1,0,0) \\ \vec{a}_4 &= (1,0,0,0,0,1,0) \\ \vec{a}_5 &= (1,0,0,0,1,0,1) \\ \vec{a}_6 &= (0,0,0,1,0,0,0). \end{aligned}$$

Chiaramente $\vec{a}_2 + \vec{a}_6 = \vec{0}$ ma non produce alcuna fattorizzazione di n, invece $\vec{a}_1 + \vec{a}_2 + \vec{a}_3 + \vec{a}_5 = \vec{0}$ produce

$$(42 \times 43 \times 61 \times 85)^2 \equiv (2 \times 3 \times 5 \times 7 \times 13)^2 \ \textit{mod}\ 1829$$

ovvero $1459^2 \equiv 901^2 \ \textit{mod}\ 1829$ da cui si ricava che $\gcd(1459 + 901, 1829) = 59$ e quindi $1829 = 31 \times 59$.

□

9.4 Altri metodi di attacco

9.4.1 Calcolo di $\varphi(n)$

Proposizione 9.10. Sia $n = pq$, con p, q primi distinti allora valgono i seguenti fatti:

1. la conoscenza della fattorizzazione n è equivalente alla conoscenza di $\varphi(n)$.
2. La complessità del calcolo di $\varphi(n)$ a partire dalla conoscenza di n è $O(\log n)$.
3. La complessità del calcolo della fattorizzazione di n a partire dalla conoscenza di $\varphi(n)$ è $O(\log^3 n)$.

Dimostrazione. Infatti, se $n = pq$, segue da

$$\varphi(n) = (p-1)(q-1) = n - (p+q) + 1$$

e quindi la complessità del calcolo di $\varphi(n)$ è $O(\log n)$.

Viceversa, noto $\varphi(n)$, allora $p + q = n - \varphi(n) + 1$, allora p, q sono le soluzioni di

$$X^2 - (n - \varphi(n) + 1)\, X + n = 0.$$

Quindi,

$$p, q = \frac{(n - \varphi(n) + 1) \pm \sqrt{(n - \varphi(n) + 1)^2 - 4n}}{2}$$

da cui si ricava che la sua complessità $O(\log^3 n)$.

□

Esempio 9.11. Siano noti $n = 84773093$ e $\varphi(n) = 84754668$. Allora

$$x^2 - 18426x + 84773093 = 0$$

ha come soluzioni $p = 8887$ e $q = 9539$.

9.4.2 L'esponente di decifratura

In questa sezione mostriamo che la conoscenza dell'esponente di decifratura d comporta la fattorizzazione di n in tempi polinomiali.

Sia $n = pq$ con p e q primi distinti e siano e e d le chiavi di cifratura e decifratura di un generico utente che utilizza il crittosistema RSA. Allora $ed - 1 = 2^s r \equiv 0 \ mod \ \varphi(n)$ (r dispari). Se w è un intero tale che $\gcd(w, n) = 1$, dal **Teorema di Eulero** discende che

$$w^{2^s r} \equiv 1 \ mod \ n.$$

Sia $t = \min\{0, ..., s\}$ tale che $w^{2^t r} \equiv 1 \ mod \ n$. Se $w^{2^{t-1} r} \not\equiv -1 \ mod \ n$ per $t > 0$, allora $w^{2^{t-1} r}$ è una radice quadrata non banale di 1 in $\mathbb{Z}_n$. Pertanto

$$1 < \gcd(w^{2^{t-1} r} + 1, n) < n,$$

cioè $\gcd(w^{2^{t-1} r} + 1, n) \in \{p, q\}$ e quindi n è fattorizzato.

Ciò non si verifica nel caso in cui w sia un intero tale che $\gcd(w, n) = 1$ e vale una delle seguenti congruenze:

1. $w^r \equiv 1 \ mod \ n$
2. $w^{2^t r} \equiv -1 \ mod \ n$ con qualche intero t tale che $0 \leq t \leq s - 1$.

Cioè non si verifica nel caso in cui w è una base rispetto alla quale n è uno pseudoprimo di Eulero. Pertanto, la probabilità di fattorizzare n corrisponde alla probabilità di trovare una base rispetto alla quale n non è uno pseudoprimo di Eulero. Tale probabilità, abbiamo visto essere maggiore uguale ad 1/2 per il **Teorema 8.26** .

Esempio 9.12. Siano $n = 89855713$, $e = 34986517$, $d = 82330933$ e il valore casuale $w = 5$. Allora

$$ed - 1 = 2^3 \times 360059073378795.$$

Quindi $s = 3$ e $r = 360059073378795$. Per $t = 1$

$$\begin{aligned} w^r \ mod\ n &\equiv 85877701 \\ w^{2r} \ mod\ n &\equiv 1, \end{aligned}$$

quindi

$$\gcd(85877700, 89855713) = 9871$$

e pertanto $n = 9103 \times 9871$.

Algoritmo 9.13. **(RSA-Factor** (n, a, b)**)**

commento: supponiamo che $ed \equiv 1 \ mod\ \varphi(n)$
scriviamo $ed - 1 = 2^s r$ con r dispari.
scegliamo un intero causale w compreso tra 1 e $n - 1$
$x \longleftarrow \gcd(w, n)$
if $1 < x < n$
then return (x)
commento: x è un fattore di n
$v \longleftarrow w^r \ mod\ n$
if $v \equiv 1 \ mod\ n$
then return ("insuccesso")
while $v \not\equiv 1 \ mod\ n$
do $\begin{cases} v_0 \longleftarrow v \\ v \longleftarrow v^2 \ mod\ n \end{cases}$
if $v_0 \equiv -1 \ mod\ n$
then return ("insuccesso")
else $\begin{cases} x \longleftarrow \gcd(v_0 + 1, n) \\ \text{return } (x) \end{cases}$
commento: x è un fattore di n

Chiaramente, l'**Algoritmo 9.13** è un $(m, 1 - \frac{1}{2^m})$-algoritmo di tipo Las vegas con complessità $O(\log^3 n)$.

9.4.3 Attacco di Wiener

In questa sezione presentiamo un attacco al crittosistema RSA dovuto a Norbert Wiener.

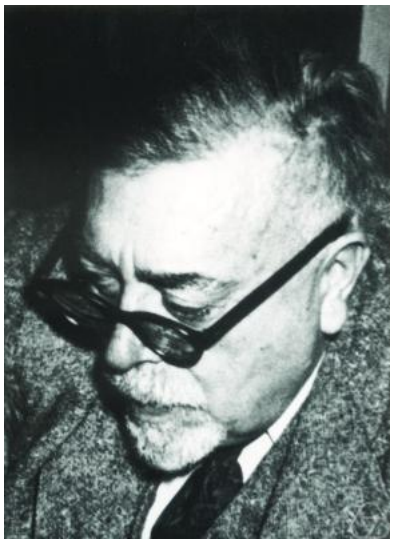


Figura 9.2: Norbert Wiener (1894 – 1964)

L'attacco permette di determinare la chiave di decifratura d nel caso in cui $n = pq$ con $q < p < 2q$ e $3d < \sqrt[4]{n}$. Siccome (n, e) è la chiave pubblica, l'idea che sta alla base è determinare d attraverso lo sviluppo in frazione continua di $\frac{e}{n}$.

Per comprendere il suddetto attacco è necessario fare una brevissima introduzione sulle frazioni continue.

Definizione 9.14. **(Frazione Continua)**
Siano $a_0, a_1, ..., a_n$ interi tali che $a_1, ..., a_n > 0$, allora un'espressione della forma

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots + a_n}}}}$$

si dice **frazione continua finita** e la si denota con $[a_0, a_1, ..., a_n]$.

Siano a, b interi positive tali che $\gcd(a,b) = 1$ e siano $(q_1, ..., q_m)$ gli interi positivi determinati attraverso l'**Algoritmo Euclideo**, ovvero, posto $r_0 = a$ e $r_1 = b$, vale che

$$\begin{array}{ll} r_0 = q_1 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_2 r_2 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{m-2} = q_{m-1} r_{m-1} + r_m & 0 < r_m < r_{m-1} \\ r_{m-1} = q_m r_m & \end{array}$$

Proposizione 9.15. Vale che

$$\frac{a}{b} = [q_1, ..., q_m].$$

Dimostrazione. Proviamo il seguente asserto per induzione su m. Possiamo supporre che $b \neq 1$. Quindi $m \geq 2$.

(1) $m = 2$: Quindi $r_0 = q_1 r_1 + r_2$ e $r_1 = q_2 r_2$. Quindi

$$\frac{a}{b} = \frac{r_0}{r_1} = \frac{q_1 r_1 + r_2}{r_1} = q_1 + \frac{r_2}{r_1} = \\ = q_1 + \frac{1}{\frac{r_1}{r_2}} = q_1 + \frac{1}{q_2} = [q_1, q_2].$$

(2) Supponiamo vero l'asserto per $m = k$ e proviamolo per $m = k + 1$. Quindi,

$$\begin{array}{ll} r_0 = q_1 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_2 r_2 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{k-2} = q_{k-1} r_{k-1} + r_k & 0 < r_k < r_{k-1} \\ r_{k-1} = q_k r_k + r_{k+1} & 0 < r_{k+1} < r_k \\ r_k = q_{k+1} r_{k+1} & \end{array}$$

Si noti che $\gcd(r_1, r_2) = \gcd(r_0, r_1) = 1$ e quindi applicando l'ipotesi induttiva a $\frac{r_1}{r_2}$ si ha che $\frac{r_1}{r_2} = [q_2, ..., q_{k+1}]$. D'altra parte, $\frac{a}{b} = \frac{r_0}{r_1} = q_1 + \frac{1}{\frac{r_1}{r_2}}$ e quindi $\frac{a}{b} = [q_1, q_2, ..., q_{k+1}]$.

□

- Se $\frac{a}{b} = [q_1, ..., q_m]$, allora $[q_1, ..., q_m]$ è detta **frazione continua** di $\frac{a}{b}$.
- Per ogni $1 \leq j \leq m$, definiamo **convergente j-esimo** il numero razionale $C_j = [q_1, ..., q_j]$.

Lemma 9.16. *Per ogni $1 \le j \le m$ il convergente j-esimo il numero razionale $C_j = \frac{c_j}{d_j}$, dove*

$$c_j = \begin{cases} 1 & se\ j = 0 \\ q_1 & se\ j = 1 \\ q_j c_{j-1} + c_{j-2} & se\ j \ge 2 \end{cases} \qquad d_j = \begin{cases} 1 & se\ j = 0 \\ 1 & se\ j = 1 \\ q_j d_{j-1} + d_{j-2} & se\ j \ge 2 \end{cases}$$

La **Proposizione 9.15** e il **Lemma 9.16** forniscono un metodo efficiente per determinare lo sviluppo di un qualsiasi razionale in frazione continua e per il calcolo dei convergenti, rispettivamente. Vediamo degli esempi.

Esempio 9.17. Determinare lo sviluppo di $\frac{34}{99}$ mediante frazione continua e le relative frazioni continue convergenti.

Attraverso l'**Algoritmo Euclideo** si ottiene

$$\begin{aligned} 34 &= 0 \times 99 + 34 \\ 99 &= 2 \times 34 + 31 \\ 34 &= 1 \times 31 + 3 \\ 31 &= 10 \times 3 + 1 \\ 3 &= 3 \times 1 \end{aligned}$$

Quindi, lo sviluppo di $\frac{34}{99}$ mediante frazione continua è

$$\frac{34}{99} = \frac{1}{2 + \frac{1}{1 + \frac{1}{10 + \frac{1}{3}}}} = [0, 2, 1, 10, 3].$$

Le frazioni convergenti convergenti sono

$$\begin{aligned} &[0] = 0 \\ &[0, 2] = 1/2 \\ &[0, 2, 1] = 1/3 \\ &[0, 2, 1, 10] = 11/32 \\ &[0, 2, 1, 10, 3] = 34/99. \end{aligned}$$

□

Esempio 9.18. Determinare lo sviluppo di $\frac{60728973}{160523347}$ mediante frazione continua e le relative frazioni continue convergenti.

Attraverso l'**Algoritmo Euclideo** si ottiene

$$\begin{aligned}
60728973 &= 0 \times 160523347 + 60728973\\
160523347 &= 2 \times 60728973 + 39\,065\,401\\
60728973 &= 1 \times 39\,065\,401 + 21\,663\,572\\
39\,065\,401 &= 1 \times 21\,663\,572 + 17\,401\,829\\
21\,663\,572 &= 1 \times 17\,401\,829 + 4261\,743\\
17\,401\,829 &= 4 \times 4261\,743 + 354\,857\\
4261\,743 &= 12 \times 354\,857 + 3459\\
354\,857 &= 102 \times 3459 + 2039\\
3459 &= 1 \times 2039 + 1420\\
2039 &= 1 \times 1420 + 619\\
1420 &= 2 \times 619 + 182\\
619 &= 3 \times 182 + 73\\
182 &= 2 \times 73 + 36\\
73 &= 2 \times 36 + 1\\
36 &= 36 \times 1
\end{aligned}$$

Quindi,

$$\frac{60728973}{160523347} = [0, 2, 1, 1, 14, 12, 102, 1, 1, 2, 3, 2, 2, 36].$$

I primi convergenti sono

$$\begin{aligned}
&[0] = 0\\
&[0, 2] = 1/2\\
&[0, 2, 1] = 1/3\\
&[0, 2, 1, 1] = 2/5\\
&[0, 2, 1, 1, 14] = 3/8\\
&[0, 2, 1, 1, 14, 12] = 14/37\\
&\vdots
\end{aligned}$$

□

Teorema 9.19. *Siano a, b, c, d interi tali che $\gcd(a, b) = \gcd(c, d) = 1$. Se*

$$\left|\frac{a}{b} - \frac{c}{d}\right| < \frac{1}{2d^2}$$

allora $\frac{c}{d}$ è uno dei convergenti di $\frac{a}{b}$.

Sia $n = pq$ con p, q primi distinti e siano e, d, dove $1 < e, d < \varphi(n)$ gli esponenti di cifratura e decifratura del generico utente che utilizza il crittosistema RSA. Sia $k \in \mathbb{Z}$ tale che $ed - 1 = k\varphi(n)$.

Teorema 9.20. *(Teorema di Wiener)*
Se $q < p < 2q$ e $3d < \sqrt[4]{n}$ allora $\frac{k}{d}$ è uno dei convergenti dello sviluppo mediante frazioni continue di $\frac{e}{n}$.

Dimostrazione. Siccome $ed - 1 = k\varphi(n)$ implica $k\varphi(n) < ed < \varphi(n)d$ e quindi $k < d$, allora

$$3k < 3d < \sqrt[4]{n}. \tag{9.3}$$

Poiché $q < p$, allora $q^2 < pq = n$ e quindi $q < \sqrt{n}$.
Inoltre, da $p < 2q$, segue che

$$0 < n - \varphi(n) = pq - (p-1)(q-1) = p + q - 1 < 2q + q - 1 < 3q < 3\sqrt{n}.$$

Da (9.3) segue

$$0 < k(n - \varphi(n)) < 3k\sqrt{n} < \sqrt[4]{n^3}. \tag{9.4}$$

Quindi,

$$\begin{aligned}
\left|\frac{e}{n} - \frac{k}{d}\right| &= \left|\frac{ed - kn}{dn}\right| = \left|\frac{ed - 1 + 1 - kn}{dn}\right| = \left|\frac{k\varphi(n) + 1 - kn}{dn}\right| \\
&= \left|\frac{1 - k(n - \varphi(n))}{dn}\right| \leq \frac{k(n - \varphi(n))}{dn} \leq \frac{\sqrt[4]{n^3}}{dn} \\
&= \frac{1}{d\sqrt[4]{n}} \leq \frac{1}{3d^2} < \frac{1}{2d^2}.
\end{aligned}$$

Pertanto, $\frac{k}{d}$ è uno dei convergenti dell'espansione mediante frazioni continue di $\frac{e}{n}$ per il **Teorema 9.19**.

□

Algoritmo 9.21. **(Wiener (n, e))**
$(q_1, ..., q_m; r_m) \longleftarrow$ **Algoritmo Euclideo** (n, b)
$c_0 \longleftarrow 1$
$c_1 \longleftarrow q_1$
$d_0 \longleftarrow 0$
$d_1 \longleftarrow 1$
for $j \longleftarrow 1$ **to** m
do
- $n' \longleftarrow (d_j e - 1)/c_j$
- **commento:** $n' = \varphi(n)$ if c_j/d_j è il convergente corretto
- **if** n' è un intero
- **then**
  - siano p e q le soluzioni dell'equazione $x^2 - (n - n' + 1)x + n = 0$
  - **if** p e q sono interi positivi minori di n
  - **then return** (p, q)
- $j \longleftarrow j + 1$
- $c_j \longleftarrow q_j c_{j-1} + c_{j-2}$
- $d_j \longleftarrow q_j d_{j-1} + d_{j-2}$

return ("Insuccesso")

Vediamo con un esempio come funziona l'attacco di Wiener.

Esempio 9.22. Siano $n = 160523347$ e $e = 60728973$, quindi $\frac{e}{n} = \frac{60728973}{160523347}$. Si ricordi che,

$$\frac{60728973}{160523347} = [0, 2, 1, 1, 14, 12, 102, 1, 1, 2, 3, 2, 2, 36]$$

e le prime frazioni convergenti sono

$$0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{14}{37}, \cdots$$

$\frac{c_j}{d_j}$	$n' = \frac{d_j e - 1}{c_j}$	$x^2 - (n - n' + 1)x + n = 0$	Fattorizzazione di n
$\frac{1}{2}$	121 457 945	$x^2 - 39\,065\,403x + 160523347 = 0$	no
$\frac{1}{3}$	182 186 918	$x^2 + 21\,663\,570x + 160523347 = 0$	no
$\frac{2}{5}$	151 822 432	$x^2 - 8700\,916x + 160523347 = 0$	no
$\frac{3}{8}$	$\frac{485\,831\,783}{3}$	-	no
$\frac{14}{37}$	160 498 000	$x^2 - 25\,348x + 160523347 = 0$	$n = 12347 \times 13001$

Si noti che

$$d = e^{-1} \ mod\ (12346 \times 13000) = 37 < \frac{\sqrt[4]{n}}{3} = 37.52.$$

□

10 Problema del Logaritmo Discreto

Il **Problema Del Logaritmo Discreto (PLD)**, che è alla base di numerosi crittosistemi che vedremo nel corso, è definito come segue:

Problema 10.1. (Logaritmo Discreto)
Istanza: Siano $(G, \cdot)$ un gruppo moltiplicativo, $\alpha \in G$ di ordine n e $\beta \in \langle \alpha \rangle$.
Domanda: Determinare l'unico intero a, con $0 \leq a \leq n-1$ tale che

$$\alpha^a = \beta.$$

Siffatto intero a si dice logaritmo discreto di β in base α e lo si denota con $\log_\alpha \beta$.

L'utilità del **PLD** in Crittografia risiede nel fatto che determinare logaritmi discreti è (probabilmente) computazionalmente difficile, mentre l'operazione inversa, ovvero l'elevamento a potenza, è calcolata efficientemente attraverso il metodo dei quadrati ripetuti. Cioè,

$$F : \{0, ..., n-1\} \longrightarrow \langle \alpha \rangle, a \longmapsto \alpha^a$$

è una funzione **one-way**.

Nel 1985, T. Elgamal propose il seguente crittosistema basato sul PLD in $(\mathbb{Z}_p^*, \cdot)$, p primo.

Figura 10.1: Taher Elgamal (1955)

Definizione 10.2. **(Crittosistema di Elgamal)**
Sia p un primo tale che il PLD in $(\mathbb{Z}_p^*, \cdot)$ è computazionalmente intrattabile e sia α un elemento primitivo di $\mathbb{Z}_p^*$. Inoltre siano:

1. $\mathcal{P} = \mathbb{Z}_p^*$;
2. $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$;
3. $\mathcal{K} = \{(p, \alpha, a, \beta) : \alpha^a = \beta\}$, dove (p, α, β) è pubblica, mentre a è segreta.
4. Per $K = (p, \alpha, a, \beta)$ ed un elemento random (segreto) $k \in \mathbb{Z}_{p-1}$ vale che

$$\begin{aligned} e_{K,k} &: \mathbb{Z}_p^* \longrightarrow \mathbb{Z}_p^* \times \mathbb{Z}_p^*, x \longmapsto (y_1, y_2) = (\alpha^k \ mod\ p, x\beta^k \ mod\ p) \\ d_K &: \mathbb{Z}_p^* \times \mathbb{Z}_p^* \longrightarrow \mathbb{Z}_p^*, (y_1, y_2) \longmapsto y_2 (y_1^a)^{-1} \ mod\ p. \end{aligned}$$

Si noti che la cifratura nel crittosistema di Elgamal è randomizzata in quanto dipende, oltra che da x, anche da k. Quindi un testo in chiaro x può essere cifrato in $p-1$ modi diversi (pari al numero degli elementi di $\mathbb{Z}_{p-1}$).

Esempio 10.3. Sia $p = 2579$, siccome $\frac{p-1}{2} = 1289$ allora $2^{1289} \equiv \left(\frac{2}{2579}\right) \ mod\ 2579$ e quindi $2^{1289} \equiv -1 \ mod\ 2579$ essendo $2579 \equiv 3 \ mod\ 8$. Pertanto, $\alpha = 2$ è un elemento primitivo di $\mathbb{Z}_{2579}^*$. Sia $a = 765$, quindi $\beta = 2^{765} \ mod\ 2579 = 949$. Quindi

$$K = (2579, 2, 756, 949)$$

Supponiamo che un utente voglia trasmettere l'unità di messaggio $x = 1299$. Sia $k = 853$ l'intero generato casualmente, allora

$$\begin{aligned} y_1 &= 2^{853} \ mod\ 2759 = 435 \\ y_2 &= 1299 \times 949^{853} \ mod\ 2759 = 2396 \\ e_{K,853}(1299) &= (435, 2396) \\ d_K(435, 2396) &= 2396 \times (435^{765})^{-1} \ mod\ 2759 = 1299 = x. \end{aligned}$$

□

Il Crittosistem di Elgamal è dimostrabilmente sicuro: infatti fonda la sicurezza sull'intrattabilità computazione del problema del logaritmo discreto in $\mathbb{Z}_p^*$. Infatti, non ci sono algoritmi aventi complessità computazionale di tipo polinomiale, se p ha circa 300 cifre e $p-1$ ha un fattore primo elevato, e se α è un elemento primitivo modulo p.

10.1 L'algoritmo di Shanks

Il seguente algoritmo, dovuto a Shanks, rappresenta un compromesso tra memory utilizzata e tempo impiegato per il calcolo del Logaritmo discreto nel generico gruppo ciclico.

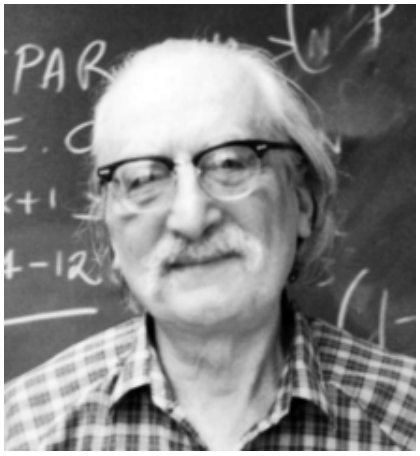


Figura 10.2: Daniel Shanks (1917 – 1996)

Sia $\alpha \in G$, con $o(\alpha) = n$ e sia $\beta \in \langle\alpha\rangle$. Allora l'**Algoritmo di Shanks** opera come segue

Algoritmo 10.4. **(Shanks**(G, n, α, β)**)**

$m \longleftarrow \lceil\sqrt{n}\rceil$

for $j \longleftarrow 0$ **to** $m-1$
do calcola α^{mj}

Crea una lista L_1 delle m coppie (j, α^{jm}) ordinate rispetto alla seconda coordinata.

for $i \longleftarrow 0$ **to** $m-1$
do calcola $\beta\alpha^{-i}$

Crea una lista L_2 delle m coppie $(i, \beta\alpha^{-i})$ ordinate rispetto alla seconda coordinata.

Cerca le coppie in L_1 e in L_2 aventi la stessa seconda coordinata. Cioè se $(j_0, y) \in L_1$ e $(i_0, y) \in L_2$.

$\log_\alpha \beta \longleftarrow j_0 m + i_0$.

Se $(j_0, y) \in L_1$ e $(i_0, y) \in L_2$, allora

$$\alpha^{j_0 m} = y = \beta\alpha^{-i_0} \Longrightarrow \beta = \alpha^{j_0 m + i_0} \Longrightarrow \log_\alpha \beta = j_0 m + i_0.$$

Viceversa, se dividiamo $0 \le \log_\alpha \beta \le n-1$ per m esistono degli interi i_0, j_0 con $0 \le i_0, j_0 \le m-1$ tali che

$$\log_\alpha \beta = j_0 m + i_0.$$

Si noti che $\sqrt{n} \le m$, allora

$$\log_\alpha \beta \le n - 1 \le m^2 - 1 = (m-1)m + m - 1$$

e quindi $j_0 \le m-1$. Pertanto l'algoritmo si Shanks ha successo se, e solo se, $\beta \in \langle\alpha\rangle$.

La complessità computazionale dell'Algoritmo di Shanks è

$$O(\lceil\sqrt{n}\rceil \ln \lceil\sqrt{n}\rceil).$$

Esempio 10.5. Determiniamo $\log_3 525$ in $\mathbb{Z}^*_{809}$.

Siccome 809 è primo, $808 = 2^3 101$ e $\left(\frac{3}{809}\right) = \left(\frac{2}{3}\right) = -1$, allora $3^{404} \equiv -1 \ mod\ 809$. Pertanto, $\alpha = 3$ è un elemento primitivo di $\mathbb{Z}^*_{809}$. Allora $n = 809$, quindi

$$m = \lceil\sqrt{809}\rceil = \lceil 28,\ 443\rceil = 29$$
$$\alpha^{29} \ mod\ 809 = 99$$

Allora calcoliamo $(j, 99^j)$ con $0 \le j \le 28$ ottenendo così

$(0,1)$	$(1,99)$	$(2,93)$	$(3,308)$	$(4,559)$
$(5,329)$	$(6,211)$	$(7,664)$	$(8,207)$	$(9,268)$
$(10,644)$	$(11,654)$	$(12,26)$	$(13,147)$	$(14,800)$
$(15,727)$	$(16,681)$	$(17,464)$	$(18,632)$	$(19,275)$
$(20,528)$	$(21,496)$	$(22,564)$	$(23,15)$	$(24,676)$
$(25,586)$	$(26,575)$	$(27,295)$	$(28,81)$	

che ordinate rispetto alla seconda coordinata producono la lista L_1.
Ora, siccome $3^{-1} \equiv 270 \ mod\ 809$ calcoliamo $(i, 525 \times 270^i)$ con $0 \le i \le 28$. Allora

$(0,525)$	$(1,175)$	$(2,328)$	$(3,379)$	$(4,396)$
$(5,132)$	$(6,44)$	$(7,554)$	$(8,724)$	$(9,511)$
$(10,440)$	$(11,686)$	$(12,768)$	$(13,256)$	$(14,355)$
$(15,388)$	$(16,399)$	$(17,133)$	$(18,314)$	$(19,644)$
$(20,754)$	$(21,521)$	$(22,713)$	$(23,777)$	$(24,259)$
$(25,356)$	$(26,658)$	$(27,489)$	$(28,163)$	

che ordinate rispetto alla seconda coordinata producono la lista L_2.
A questo punto si vede che $(10,644) \in L_1$ e $(19,644) \in L_2$, quindi

$$\log_3 525 = (29 \times 10 + 19) \ mod\ 809 = 309.$$

E' facile vedere che effettivamente $3^{309} \equiv 525 \ mod\ 809$.

□

10.2 L'algoritmo Rho di Pollard

Siano $(G,\cdot)$ un gruppo $\alpha \in G$ di ordine n e sia $\beta \in \langle\alpha\rangle$. Siccome $\langle\alpha\rangle$ è ciclico di ordine n, allora possiamo considerare $\log_\alpha \beta$ come un elemento di $\mathbb{Z}_n$.

Sia $\{S_1, S_2, S_3\}$ un partizione di G in parti della stessa grandezza circa. Quindi, $G = S_1 \biguplus S_2 \biguplus S_3$. Sia

$$\begin{aligned} f : \langle\alpha\rangle \times \mathbb{Z}_n \times \mathbb{Z}_n &\longrightarrow \langle\alpha\rangle \times \mathbb{Z}_n \times \mathbb{Z}_n, \\ (x,a,b) &\longmapsto \begin{cases} (\beta x, a, b+1) & \text{se } x \in S_1 \\ (x^2, 2a, 2b) & \text{se } x \in S_2 \\ (\alpha x, a+1, b) & \text{se } x \in S_3. \end{cases} \end{aligned} \tag{10.1}$$

Lemma 10.6. *Se (x,a,b) è tale che $x = \alpha^a\beta^b$, allora $(x',a',b') = f(x,a,b)$ è tale che $x' = \alpha^{a'}\beta^{b'}$*

Dimostrazione. Vale che

$$\begin{aligned} x \in S_1 &\Longrightarrow \beta x = \beta(\alpha^a\beta^b) = \alpha^a\beta^{b+1} \\ x \in S_2 &\Longrightarrow x^2 = (\alpha^a\beta^b)(\alpha^a\beta^b) = \alpha^{2a}\beta^{2b} \\ x \in S_3 &\Longrightarrow \alpha x = \alpha(\alpha^a\beta^b) = \alpha^{a+1}\beta^b. \end{aligned}$$

Pertanto, l'asserto.

□

Quindi, definiamo per ricorrenza

$$(x_i,a_i,b_i) = \begin{cases} (1,0,0) & \text{se } i = 0 \\ f(x_{i-1},a_{i-1},b_{i-1}) & \text{se } i \geq 1 \end{cases} \tag{10.2}$$

Osservazione 10.7.

- Si noti che $1 = \alpha^0\beta^0$, quindi, per il **Lemma 10.6**, $x_i = \alpha^{a_i}\beta^{b_i}$ per ogni $i \geq 1$.
- Si noti che, per ipotesi $1 \notin S_2$, altrimenti la successione (10.2) diventa costante di costante valore $(1,0,0)$.

Lemma 10.8. *Se (x_i,a_i,b_i) e (x_j,a_j,b_j) sono tali che soddisfano $x_i = x_j$ per ogni $0 \leq i < j \leq n-1$, allora esiste $0 \leq i' \leq n-1$ tale che $(x_{i'},a_{i'},b_{i'})$, $(x_{2i'},a_{2i'},b_{2i'})$ soddisfano $x_{i'} = x_{2i'}$.*

Dimostrazione. Segue dalla definizione di f in (10.1) che se $x_i = x_j$, allora

$$x_{i+\delta} = x_{j+\delta} \text{ per ogni } \delta \in \mathbb{N}. \tag{10.3}$$

Sia $\ell = j - i$, poichè $\{i, i+1, \cdots, j-1\}$ è un sistema completo di residui modulo ℓ, esiste $i' \in \{i, i+1, \cdots, j-1\}$ tale che $i' = k_0\ell$. Sia $\delta = i' - i$, allora vale

$$x_{i'} = x_{i'+\ell}.$$

Ora applicando (10.3) con i' al posto di i, $i' + \ell$ al posto di j ed ℓ al posto di δ, si ha $x_{i'+\ell} = x_{i'+2\ell}$. Pertanto, $x_{i'} = x_{i'+2\ell}$ e, induttivamente,

$$x_{i'} = x_{j'} \text{ con } j' = i' + k\ell, k \in \mathbb{N}.$$

Se $k = k_0$, allora $j' = 2i'$ e quindi vale

$$x_{i'} = x_{2i'}.$$

□

Lemma 10.9. *Se (x_i, a_i, b_i) e (x_{2i}, a_{2i}, b_{2i}) sono tali che soddisfano $x_i = x_{2i}$ e $\gcd(b_{2i} - b_i, n) = 1$, allora*

$$\log_\alpha \beta = (a_i - a_{2i})\,(b_{2i} - b_i)^{-1} \ \ mod\ n.$$

Dimostrazione. Sia $c = \log_\alpha \beta$. Se $x_i = x_{2i}$, allora $\alpha^{a_i}\beta^{b_i} = \alpha^{a_{2i}}\beta^{b_{2i}}$ e quindi $\alpha^{a_i + cb_i} = \alpha^{a_{2i} + cb_{2i}}$. Pertanto,

$$a_i + cb_i \equiv a_{2i} + cb_{2i} \ \ mod\ n.$$

da cui si ricava $c \equiv (a_i - a_{2i})\,(b_{2i} - b_i)^{-1} \ \ mod\ n$, essendo $\gcd(b_{2i} - b_i, n) = 1$. □

Abbiamo provato che se si determinano due terne (x_i, a_i, b_i) e (x_{2i}, a_{2i}, b_{2i}) tali che $x_i = x_{2i}$ e $\gcd(b_{2i} - b_i, n) = 1$, allora si determina il logaritmo discreto.

Il caso in cui $\gcd(b_{2i} - b_i, n) > 1$ non è così intrattabile.
Infatti se $\gcd(b_{2i} - b_i, n) = d$ allora

$$c(b_{2i} - b_i) \equiv (a_i - a_{2i}) \ \ mod\ n$$

ha esattamente d possibili soluzioni. Se d non è troppo grande, è relativamente facile trovarle e vedere quale tra queste è quella che rappresenta il logaritmo discreto.

Esempio 10.10. L'intero $p = 809$ e si vede facilmente che $\alpha = 89$ ha ordine $n = 101$ in $\mathbb{Z}^*_{809}$. Inoltre $\beta = 618$ appartiene a $\langle\alpha\rangle$. Vogliamo, quindi, determinare $\log_{89} 618$.

Definiamo gli insiemi

$$\begin{aligned} S_1 &= \{x \in \mathbb{Z}^*_{809} : x \equiv 1 \ \ mod\ 3\} \\ S_2 &= \{x \in \mathbb{Z}^*_{809} : x \equiv 0 \ \ mod\ 3\} \\ S_3 &= \{x \in \mathbb{Z}^*_{809} : x \equiv 2 \ \ mod\ 3\} \end{aligned}$$

e consideriamo la $f : \langle\alpha\rangle \times \mathbb{Z}_{101} \times \mathbb{Z}_{101} \longrightarrow \langle\alpha\rangle \times \mathbb{Z}_{101} \times \mathbb{Z}_{101}$ definita come in (10.1).

Allora

i	(x_i, a_i, b_i)	(x_{2i}, a_{2i}, b_{2i})
1	$(618, 0, 1)$	$(76, 0, 2)$
2	$(76, 0, 2)$	$(113, 0, 4)$
3	$(46, 0, 3)$	$(488, 1, 5)$
4	$(113, 0, 4)$	$(605, 4, 10)$
5	$(349, 1, 4)$	$(422, 5, 11)$
6	$(488, 1, 5)$	$(683, 7, 11)$
7	$(555, 2, 5)$	$(451, 8, 12)$
8	$(605, 4, 10)$	$(344, 9, 13)$
9	$(451, 5, 10)$	$(112, 11, 13)$
10	$(422, 5, 11)$	$(422, 11, 15)$

La prima collisione del tipo $x_h = x_{2h}$ è $x_{10} = x_{20} = 422$, allora

$$\log_{89} 618 = (5-11)(15-11)^{-1} \ mod\ 101 = (-6 \times 76) \ mod\ 101 = 49.$$

Pertanto, $\log_{89} 618 = 49$ in $\mathbb{Z}^*_{809}$. Infatti $89^{49} \equiv 618 \ mod\ 809$.

□

Supponiamo come sempre che $(G, \cdot)$ un gruppo $\alpha \in G$ di ordine n e sia $\beta \in \langle \alpha \rangle$.

Algoritmo 10.11. (Rho di Pollard per il Logaritmo Discreto(G, n, α, β))

Procedura $f(x, a, b)$
if $x \in S_1$
then $f \longleftarrow (\beta \cdot x, a, (b+1) \ mod\ n)$
else if $x \in S_2$
then $f \longleftarrow (x^2, (2 \cdot a) \ mod\ n, (2 \cdot b) \ mod\ n)$
else $f \longleftarrow (\alpha \cdot x, (a+1) \ mod\ n, b)$
return (f)
main
definiamo la partizione $\{S_1, S_2, S_3\}$ di G
$(x, a, b) \longleftarrow (1, 0, 0)$
$(x', a', b') \longleftarrow f(x, a, b)$
while $x \neq x'$
do $\begin{cases} (x, a, b) \longleftarrow f(x, a, b) \\ (x', a', b') \longleftarrow f(x', a', b') \\ (x', a', b') \longleftarrow f(x', a', b') \end{cases}$
if $\gcd(b' - b, n) \neq 1$
then return ("Insuccesso)
else return $((a - a')(b' - b)^{-1} \ mod\ n)$

Si prova che, sotto opportune ipotesi relative alla casualità di f, l'**Algoritmo 10.11** ha compessità $O(\sqrt{n})$.

10.3 L'algoritmo di Pohlig-Hellmann

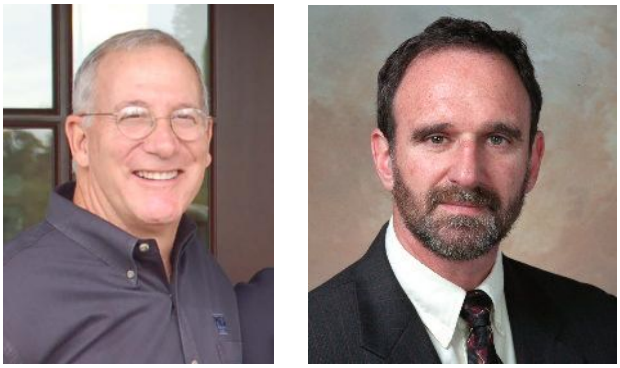


Figura 10.3: Stephen Pohlig e Martin Edward Hellman

Siano $(G,\cdot)$ e α un suo elemento di ordine n. Sia ora $\beta \in \langle\alpha\rangle$ il valore $c = \log_\alpha \beta$ è univocamente determinato modulo n. Infatti, $\alpha^{c+hn} = \alpha^c = \beta$. Pertanto, ai fini della determinazione del logaritmo discreto è sufficiente determinare $c \ mod \ n$.

Sia

$$n = \prod_{i=1}^{\ell} p_i^{k_i}$$

la fattorizzazione completa di n. Per determinare $c \ mod \ n$ grazie al **Teorema Cinese dei Resti** è sufficiente determinare $c \ mod \ p_i^{k_i}$ per ogni $i = 1, .., \ell$. Quindi si p uno dei divisori primi di n e sia p^k la massima potenza di p che divide n.

Vogliamo determinare

$$x \equiv c \ mod \ p^k$$

dove $0 \le x \le p^k - 1$. Rappresentiamo x in base p. Allora

$$x = \sum_{i=0}^{k-1} x_i p^i, \text{ dove } 0 \le x_i \le p-1 \text{ per } 0 \le i \le k-1.$$

Pertanto l'obbiettivo è determinare tutti gli x_i (e quindi x). Per fare ciò esprimiamo c come segue

$$c = x + \mu p^k, \quad \mu \in \mathbb{Z}$$

Quindi,

$$c = \sum_{j=0}^{k-1} x_j p^j + \mu p^k.$$

Lemma 10.12. *Sia*

$$\beta_j = \begin{cases} \beta & \text{se } j = 0 \\ \beta\alpha^{-(x_0+x_1p+\cdots x_{j-1}p^{j-1})} & \text{se } 1 \le j \le k-1 \end{cases}$$

allora

$$\beta_j^{n/p^{j+1}} = \alpha^{x_j n/p} \text{ per } 0 \le j \le k-1. \tag{10.4}$$

Dimostrazione. Se $j = 0$, risulta

$$\begin{aligned} \beta_j^{n/p^{j+1}} &= \beta^{n/p} = (\alpha^c)^{n/p} = \left(\alpha^{x_0+x_1p+\cdots+x_{k-1}p^{k-1}+\mu p^k}\right)^{n/p} = \\ &= \left(\alpha^{x_0+M_0p}\right)^{n/p} = (\alpha^{x_0})^{n/p} + \left(\alpha^{M_0p}\right)^{n/p} = \alpha^{x_0 n/p}\alpha^{M_0 n} = \alpha^{x_0 n/p}. \end{aligned}$$

Se $j \ge 1$, vale che

$$\begin{aligned} \beta_j^{n/p^{j+1}} &= \left(\beta\alpha^{-(x_0+x_1p+\cdots+x_{j-1}p^{j-1})}\right)^{n/p^{j+1}} = \left(\alpha^c\alpha^{-(x_0+x_1p+\cdots+x_{j-1}p^{j-1})}\right)^{n/p^{j+1}} = \\ &= \left(\alpha^{c-(x_0+x_1p+\cdots x_{j-1}p^{j-1})}\right)^{n/p^{j+1}} = \left(\alpha^{x_jp^j+\cdots x_{k-1}p^{k-1}+\mu p^k}\right)^{n/p^{j+1}} = \\ &= \left(\alpha^{x_jp^j+M_jp^{j+1}}\right)^{n/p^{j+1}} = \left(\alpha^{x_jp^j}\right)^{n/p^{j+1}}\left(\alpha^{M_jp^{j+1}}\right)^{n/p^{j+1}} = \\ &= \alpha^{x_j n/p}\alpha^{M_j n} = \alpha^{x_j n/p}. \end{aligned}$$

□

Come diretta conseguenza del **Lemma 10.12** si ha

$$\beta_{j+1} = \beta_j\alpha^{-x_jp^j} \text{ per } 1 \le j \le k-1. \tag{10.5}$$

Sulla base delle precedenti osservazioni, l'**Algoritmo di Pohlig-Hellmann** opera come segue

1. Determina x_0 a partire da (10.4) per $j = 0$, ovvero da $\beta^{n/p} = \beta_0^{n/p} = \alpha^{x_0 n/p}$. Partendo da $\gamma = \alpha^{n/p}$ si calcola $\gamma^2, \gamma^3, \ldots$ fino a determinare $i \le p-1$ tale che $\gamma^i = \beta^{n/p}$. Quando questo accade, allora $x_0 = i$ (si noti che $x_0 = \log_{\alpha^{n/p}} \beta^{n/p}$).
2. Si calcola attraverso (10.5), ovvero $\beta_1 = \beta_0\alpha^{-x_0}$ (x_0 è stato determinato nel passo precedente!).
3. Siccome da (10.4) vale che $\beta_1^{n/p^2} = \alpha^{x_1 n/p}$ allora iniziando da $\gamma = \alpha^{n/p}$ si calcola $\gamma^2, \gamma^3, \ldots$ fino a determinare $i' \le p-1$ tale che $\gamma^{i'} = \beta_1^{n/p^2}$. Quando questo accade, allora $x_1 = i'$, dove $x_1 = \log_{\alpha^{n/p}} \beta^{n/p^2}$.

Si procede iterativamente determinando

$$x_0, \beta_1, x_1, \beta_2, ..., \beta_{k-1}, x_{k-1}$$

applicando in modo alternato (10.4) e (10.5). Alla fine del procedimento si conoscono tutti gli x_i (e quindi x). Quindi

$$c \equiv \sum_{j=0}^{k-1} x_j p^j \ \ mod\ p^k.$$

Applicando lo stesso procedimento a tutti i divisori primi di n si giunge al seguente sistema congruenziale

$$\begin{cases} c \equiv \sum_{j=0}^{k_1-1} x_{j1} p_1^j \ \ mod\ p_1^k \\ \vdots \\ c \equiv \sum_{j=0}^{k_\ell-1} x_{j\ell} p_\ell^j \ \ mod\ p_\ell^k \end{cases}$$

e quindi a $c\ mod\ n$ per il **Teorema Cinese dei Resti**.

Remark 10.13. Il punto chiave su cui si basa l'efficienza dell'algoritmo è che il calcolo degli $x_{ji} = \log_{\alpha^{n/p_i}} \beta^{n/p_i^j}$ dove $0 \le x_{ji} \le p_i - 1$ deve essere piuttosto spedito. Ne segue che p_j non deve essere elevato. Pertanto n è prodotto di primi piccoli, ovvero è **piatto**.

Algoritmo 10.14. **(Pohlig-Hellmann**$(G, n, \alpha, \beta, p, k)$**)**

$j \longleftarrow 0$
$\beta_j \longleftarrow \beta$
while $j \le k-1$
do $\begin{cases} \delta \longleftarrow \beta_j^{n/p^{j+1}} \\ \text{determina } i \text{ tale che } \delta = \alpha^{in/p} \\ x_j \longleftarrow i \\ \beta_{j+1} \longleftarrow \beta_j \alpha^{-x_j p^j} \\ j \longleftarrow j+1 \end{cases}$
return $(x_0, ..., x_{k-1})$

Esempio 10.15. Supponiamo che $p = 29$ e $\alpha = 2$. E' facile vedere che α è un elemento primitivo di $\mathbb{Z}_{29}^*$. Allora $n = p - 1 = 28 = 2^2 \times 7$.

Sia $\beta = 18$ vogliamo determinare $c = \log_2 18$ in $\mathbb{Z}_{29}^*$.

1. Determiniamo $c \ mod\ 4$ e $c \ mod\ 7$ e da qui determiniamo $c \ mod\ 28$.
2. Siccome $c \equiv (x_{01} + 2x_{11}) \ mod\ 4$, da (10.4) segue che $18^{14} \equiv 2^{14x_{01}} \ mod\ 29$ con $0 \leq x_{01} \leq 1$. Si vede facilmente che $x_{01} = 1$.
3. Ora da (10.5) segue che $\beta_1 = 18 \cdot 2^{-1} \equiv 19 \ mod\ 9$.
4. Da (10.4) si ha $19^7 \equiv 2^{x_{11}14} \ mod\ 29$ con $0 \leq x_{11} \leq 1$ che in realtà produce $x_{11} = 1$.

Pertanto $c \equiv 3 \ mod\ 4$.
Siccome $c \equiv x_{02} \ mod\ 7$, da (10.4) segue che $18^4 \equiv 2^{4x_{02}} \ mod\ 29$ con $0 \leq x_{01} \leq 6$. Si vede facilmente che $x_{02} = 4$ e quindi $c \equiv 4 \ mod\ 7$. Allora

$$\begin{cases} c \equiv 3 \ mod\ 4 \\ c \equiv 4 \ mod\ 7 \end{cases}$$

Per il **Teorema Cinese dei Resti** vale che $c \equiv 11 \ mod\ 28$. Quindi, $\log_2 18 = 11$ in $\mathbb{Z}_{29}^*$.

□

Nell'**Algoritmo 10.14** ci sono al più c iterazioni ed in ognuna di esse si calcola $\log_{\alpha^{n/p}} \beta^{n/p^j}$ che ha complessità $O(\sqrt{n} \ln \sqrt{n})$ usando, per esempio, l'algoritmo di Shanks. Pertanto, la complessità dell'**Algoritmo 10.14** è $O(k\sqrt{n} \ln \sqrt{n})$.

10.4 Il Metodo del Calcolo dell'Indice

Gli algoritmi visti precedentemente si utilizzavano per il calcolo del logaritmo discreto in un qualsiasi gruppo finito. L'algoritmo che descriviamo ora, noto come **Metodo del Calcolo dell'Indice**, si utilizza per il calcolo del logaritmo discreto in $\mathbb{Z}_p^*$ con p primo. Il Metodo del Calcolo dell'Indice è più veloce degli altri algoritmi per il calcolo del logaritmo discreto in $\mathbb{Z}_p^*$.

Siano $\alpha, \beta \in \mathbb{Z}_p^*$, con α un elemento primitivo di $\mathbb{Z}_p^*$, e sia $\mathcal{B} = \{p_1, ..., p_B\}$ un insieme di primi "piccoli", allora l'algoritmo consiste in due passi:

1. Calcolo dei logaritmi discreti $\log_\alpha p_1, ..., \log_\alpha p_B$
2. Calcolo dei logaritmo discreto $\log_\alpha \beta$ attraverso $\log_\alpha p_1, ..., \log_\alpha p_B$.

Sia $C > B$, per esempio $C = B + 10$ e si costruiscono C congruenze modulo p, che hanno la seguente forma:

$$\alpha^{x_j} \equiv p_1^{\alpha_{1j}} p_2^{\alpha_{2j}} \cdots p_B^{\alpha_{Bj}} \ mod\ p. \tag{10.6}$$

Per $1 \leq j \leq C$ (per esempio gli x_j possono essere determinati in modo casuale). Siccome $\alpha^{\log_\alpha p_i} = p_i$, allora

$$\alpha^{x_j} \equiv \alpha^{\alpha_{1j}\log_\alpha p_1 + \alpha_{2j}\log_\alpha p_2 + \cdots \alpha_B \log_\alpha p_B} \ mod\ p$$

e quindi (10.6) è equivalente a

$$x_j \equiv (\alpha_{1j}\log_\alpha p_1 + \alpha_{2j}\log_\alpha p_2 + \cdots \alpha_{Bj}\log_\alpha p_B) \ mod\ (p-1)\,.$$

Se sistema congruenziale nelle variabili $\log_\alpha p_1, ..., \log_\alpha p_B$

$$\begin{cases} \alpha_{11}\log_\alpha p_1 + \alpha_{21}\log_\alpha p_2 + \cdots \alpha_{B1}\log_\alpha p_B \equiv x_1 \ mod\ (p-1) \\ \vdots \\ \alpha_{1C}\log_\alpha p_1 + \alpha_{2C}\log_\alpha p_2 + \cdots \alpha_{BC}\log_\alpha p_B \equiv x_C \ mod\ (p-1)\,. \end{cases}$$

ha una soluzione, questa viene memorizzata. Una volta fatto ciò il calcolo di $\log_\alpha \beta$ avviene attraverso un algoritmo randomizzato di tipo Las Végas: si genera un intero casuale s tale che $1 \leq s \leq p-1$ e si calcola $\gamma = \beta\alpha^s \ mod\ p$, lo si cerca di fattorizzare rispetto alla base $\mathcal{B}$, ovvero si cerca di scrivere

$$\beta\alpha^s \equiv p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B} \ mod\ p,$$

o equivalentemente,

$$\log_\alpha \beta \equiv (c_1\log_\alpha p_1 + c_2\log_\alpha p_2 + \cdots c_B \log_\alpha p_B - s) \ mod\ (p-1).$$

Siccome, tutti i termini al secondo membro sono noti, è così determinato $\log_\alpha \beta$.

Esempio 10.16. Sia $p = 10007$ un primo, sia $\alpha = 5$ un elemento primitivo di $\mathbb{Z}^*_{10007}$ e sia $\mathcal{B} = \{2, 3, 5, 7\}$. Supponiamo che **alcuni** degli interi casuali generati siano $x_1 = 4063$, $x_2 = 5136$ e $x_3 = 9865$. Allora

$$\begin{cases} 5^{4063} \ mod\ 10007 = 42 = 2 \times 3 \times 7 \\ 5^{5136} \ mod\ 10007 = 54 = 2 \times 3^3 \\ 5^{9865} \ mod\ 1007 = 189 = 3^3 \times 7 \end{cases}$$

che è equivalente a

$$\begin{cases} \log_5 2 + \log_5 3 + \log_5 7 = 4063 \ mod\ 10006 \\ \log_5 2 + 3\log_5 3 = 5136 \ mod\ 10006 \\ 3\log_5 3 + \log_5 7 = 9865 \ mod\ 10006 \end{cases}$$

da cui si ricava

$$\log_5 2 = 6578 \quad \log_5 3 = 6190 \quad \log_5 7 = 1301.$$

Supponiamo di voler determinare $\log_5 9451$. Si sceglie un esponente "casuale" $s = 7736$ e si calcola

$$\gamma = \left(9451 \times 5^{7736}\right) \ mod\ 10007 = 8400 = 2^4 3^1 5^2 7^1,$$

quindi

$$\begin{aligned} \log_5 9451 &\equiv \left(4\log_5 2 + \log_5 3 + 2\log_5 5 + \log_5 7\right) \ mod\ 10006 \\ &\equiv \left(4 \times 6578 + 6190 + 2 \times 1 + 1301 - 7736\right) \ mod\ 10006 \\ &\equiv 6057. \end{aligned}$$

È facile verificare che effettivamente $5^{6057} \equiv 9451 \ mod\ 10007$.

□

11 Curve Ellittiche

11.1 Curve Algebriche

Sia $\mathbb{K}$ un campo, $\left(\mathbb{K}^3\right)^* = \mathbb{K}^3 \setminus \{(0,0,0)\}$ e si consideri lo spazio vettoriale $\left(\mathbb{K}^3, +, \cdot\right)$. Siano $(x_1, x_2, x_3), (x_1', x_2', x_3')$ due elementi di $\left(\mathbb{K}^3\right)^*$, definiamo la seguente relazione binaria:

$$(x_1, x_2, x_3) \sim (x_1', x_2', x_3') \iff \exists \lambda \in \mathbb{K} \setminus \{0\} \text{ tale che } (x_1, x_2, x_3) = \lambda (x_1', x_2', x_3').$$

Chiaramente $\sim$ è una relazione di equivalenza su $\left(\mathbb{K}^3\right)^*$ e l'insieme quoziente $\left(\mathbb{K}^3\right)^* / \sim$, che si denota con $PG(2, \mathbb{K})$, si dice **piano proiettivo sul campo** $\mathbb{K}$.

Il generico elemento di $PG(2, \mathbb{K})$, detto **punto**, è:

$$[(x_1, x_2, x_3)]_\sim = \{\lambda (x_1, x_2, x_3) \text{ t.c. } \lambda \in \mathbb{K} \setminus \{0\}\}.$$

Chiaramente i punti di $PG(2, \mathbb{K})$ sono i sottospazi 1-dimensionali di $\mathbb{K}^3$ privati del vettore nullo. Per semplicità, un punto P di $PG(2, \mathbb{K})$ verrà indicato con un suo rappresentante, cioè se P è individuato da $[(x_1, x_2, x_3)]_\sim$ allora scriveremo $P = (x_1, x_2, x_3)$ tenendo presente che la terna (x_1, x_2, x_3) è data a meno di un fattore di proporzionalità non nullo. La terna (x_1, x_2, x_3) rappresenta le **coordinate proiettive omogenee** di P.

Definiamo **rette** di $PG(2, \mathbb{K})$ i sottospazi 2-dimensionali (piani vettoriali) di $\mathbb{K}^3$ privati del vettore nullo. Ogni piano vettoriale ha chiaramente equazione $ax_1 + bx_2 + cx_3 = 0$ dove $(a, b, c) \in \left(\mathbb{K}^3\right)^*$ sono individuati a meno di un fattore di proporzionalità non nullo. Pertanto, possiamo rappresentare la generica retta di $PG(2, \mathbb{K})$ con la scrittura $[a, b, c]$ a meno di un fattore di proporzionalità non nullo. La terna $[a, b, c]$ rappresenta le **coordinate proiettive omogenee** della generica retta di $PG(2, \mathbb{K})$.

I punti di $PG(2, \mathbb{K})$ che giacciono sulla **retta impropria** r_∞: $x_3 = 0$ sono detti **punti impropri**. I punti di $PG(2, \mathbb{K})$ che hanno $x_3 \neq 0$ sono detti **punti propri**.

L'insieme $AG(2, \mathbb{K}) = PG(2, \mathbb{K}) \setminus r_\infty$, è detto **piano affine**. Il generico punto proprio $Q = (x_1, x_2, x_3)$ $(x_3 \neq 0)$ corrisponde al punto di $AG(2, \mathbb{K})$ di coordinate $(x, y) = \left(\frac{x_1}{x_3}, \frac{x_2}{x_3}\right)$.

Definizione 11.1. **(Curva Algebrica Piana di ordine n)**
Si dice **curva algebrica piana di ordine** n, l'insieme dei punti di $PG(2,\mathbb{K})$ le cui coordinate proiettive verificano un'equazione del tipo $F(x_1,x_2,x_3)=0$ dove $F(x_1,x_2,x_3)$ è un polinomio omogeneo di grado n a coefficienti in $\mathbb{K}$, nelle variabili x_1,x_2,x_3. In generale una curva algebrica di ordine n si indica con $\mathcal{C}^n$ e $F(x_1,x_2,x_3)=0$ si dice **equazione di $\mathcal{C}^n$ in coordinate proiettive omogenee**.

Ricordiamo che un polinomio $F(x_1,x_2,x_3)$ di grado n è omogeneo se e solo se $F(\lambda x_1,\lambda x_2,\lambda x_3)=\lambda^n F(x_1,x_2,x_3)$ per ogni $\lambda\in\mathbb{K},\lambda\neq 0$.

Le curve algebriche $\mathcal{C}^1$, $\mathcal{C}^2$, $\mathcal{C}^3$ sono le rette, le coniche, le cubiche, rispettivamente:

$$\mathcal{C}^1 \;:\; a_1x_1+a_2x_2+a_3x_3=0;$$

$$\mathcal{C}^2 \;:\; a_1x_1^2+a_2x_2^2+a_3x_3^2+a_4x_1x_2+a_5x_1x_3+a_6x_2x_3=0;$$

$$\mathcal{C}^3 \;:\; a_1x_1^3+a_2x_2^3+a_3x_3^3+a_4x_1x_2x_3+a_5x_1^2x_2+a_6x_1x_2^2+ a_7x_1^2x_3+a_8x_1x_3^2+a_9x_2^2x_3+a_{10}x_2x_3^2=0.$$

In seguito, limiteremo la nostra attenzione alle sole cubiche.

Definizione 11.2. **(Punto semplice e doppio)**
Un punto P_0 di una cubica $\mathcal{C}^3$ si dice **punto semplice** se ogni retta per P_0, eccetto una retta, ha una sola intersezione con $\mathcal{C}^3$ in P_0. La retta che fa eccezione ha più di una intersezione con $\mathcal{C}^3$ in P_0 e si chiama **tangente principale**.

P_0 si dice **punto doppio** di $\mathcal{C}^3$ se ogni retta per P_0, eccetto due rette, ha due intersezioni con $\mathcal{C}^3$ in P_0. Le rette che fanno eccezione hanno più di due intersezioni con $\mathcal{C}^3$ in P_0 e si chiamano **tangenti principali** in P_0 a $\mathcal{C}^3$.

Proposizione 11.3. Sia $\mathcal{C}^3$ una cubica di equazione $F(x_1,x_2,x_3)=0$ e sia $P_0=\left(x_1^0,x_2^0,x_3^0\right)$ un suo punto. Indicate con F_i ed F_{ij} $(i,j=1,2,3)$ le derivate parziali formali prime e seconde di $F(x_1,x_2,x_3)$ rispettivamente, allora

P_0 è semplice $\Leftrightarrow$ $\left(F_1^0,F_2^0,F_3^0\right)\neq(0,0,0)$ e l'equazione della tangente principale in P_0 è $F_1^0x_1+F_2^0x_2+F_3^0x_3=0$;

P_0 è doppio $\Leftrightarrow$ $F_1^0=F_2^0=F_3^0=0$ e almeno una tra le $F_{ij}^0\neq 0$. In tal caso, l'equazione complessiva delle tangenti principali è una conica degenere di equazione $\sum_{i,j=1}^3 F_{ij}^0x_ix_j=0$.

Siano $\mathcal{C}^3$ una cubica su un campo $\mathbb{K}$ e P_0 un suo punto.

- Se P_0 è un punto doppio, allora l'equazione complessiva delle tangenti principali a $\mathcal{C}^3$ in P_0 consiste dell'unione di due rette ℓ_1 ed ℓ_2. Risulta che il punto P_0 si dice **cuspide** se ℓ_1 ed ℓ_2 sono coincidenti, **nodo** se ℓ_1 ed ℓ_2 sono distinte ed hanno coefficienti in $\mathbb{K}$, **punto doppio isolato** se ℓ_1 ed ℓ_2 sono distinte ed hanno coefficienti in un'estensione quadratica di $\mathbb{K}$.
- Se P_0 è un punto semplice, allora esso si dice **ordinario** se la tangente in P_0 ha esattamente due intersezioni con $\mathcal{C}^3$ in P_0 e **flesso di prima specie** se la tangente ha esattamente tre intersezioni con $\mathcal{C}^3$ in P_0.

Definizione 11.4. Una cubica $\mathcal{C}^3$ di equazione in coordinate omogenee $F(x_1, x_2, x_3) = 0$ si dice **riducibile** se il polinomio si scrive nella forma

$$F(x_1, x_2, x_3) = G_1(x_1, x_2, x_3) \cdot G_2(x_1, x_2, x_3)$$

con $G_1(x_1, x_2, x_3)$ e $G_2(x_1, x_2, x_3)$ polinomi di grado 1 e 2, rispettivamente.

Se poniamo $\mathcal{C}_1 : G_1(x_1, x_2, x_3) = 0$ e $\mathcal{C}_2 : G_2(x_1, x_2, x_3) = 0$, diremo che la curva è l'unione delle sue componenti $\mathcal{C}_1$ e $\mathcal{C}_2$.

Osservazione 11.5. Sia $F(x_1, x_2, x_3) = 0$ l'equazione in coordinate omogenee di una cubica $\mathcal{C}^3$. Allora

$$\mathcal{C}^3 \cap AG(2, \mathbb{K}) = \left\{(x_1, x_2, x_3) \in \mathcal{C}^3 : x_3 \neq 0\right\}.$$

Pertanto, per tali punti, vale che

$$\begin{aligned} F(x_1, x_2, x_3) &= \textstyle\sum_{0 \leq p+q \leq 3} a_{pq} x_1^p x_2^q x_3^{3-(p+q)} = \\ &= \textstyle\sum_{0 \leq p+q \leq 3} a_{pq} x_3^3 \left(x_1^p x_2^q x_3^{-(p+q)}\right) = \\ &= x_3^3 \textstyle\sum_{0 \leq p+q \leq 3} a_{pq} \left(\frac{x_1}{x_3}\right)^p \left(\frac{x_2}{x_3}\right)^q = \\ &= x_3^3 f\left(\tfrac{x_1}{x_3}, \tfrac{x_2}{x_3}\right) = \\ &= x_3^3 f(x, y). \end{aligned}$$

Quindi $f(x, y) = 0$ si dice **equazione affine della cubica $\mathcal{C}^3$**.

Ora esprimiamo analiticamente i concetti di punto semplice o punto doppio nel caso affine.

Proposizione 11.6. Sia $\mathcal{C}^3$ una cubica di equazione $f(x,y)=0$ in coordinate affini e sia $P_0=(x_0,y_0)$ un suo punto. Allora

P_0 è semplice $\Leftrightarrow$ $\left(f_x^0, f_y^0\right) \neq (0,0)$ e l'equazione della tangente principale in P_0 è $f_x^0(x-x_0)+f_y^0(y-y_0)=0$;

P_0 è doppio $\Leftrightarrow$ $f_x^0=f_y^0=0$ e almeno una tra le derivate parziali seconde di f calcolate in (x_0,y_0) è diversa da 0. In tal caso, l'equazione complessiva delle tangenti principali è $f_{xx}^0(x-x_0)^2+2f_{xy}^0(x-x_0)(y-y_0)+ +f_{yy}^0(y-y_0)^2=0$.

Definizione 11.7. Una cubica $\mathcal{C}^3$ si dice **non singolare** se e solo se tutti i suoi punti sono semplici.

Definizione 11.8. Una cubica $\mathcal{C}^3$ di equazione $f(x,y)=0$ in coordinate affini si dice **riducibile** se

$$f(x,y)=g_1(x,y)\cdot g_2(x,y)$$

con $g_1(x,y)$ e $g_2(x,y)$ polinomi di grado 1 e 2, rispettivamente.

Teorema 11.9. *Una cubica irriducibile $\mathcal{C}^3$ ha al più un punto doppio.*

Dimostrazione. Supponiamo per assurdo che $\mathcal{C}^3$ abbia almeno due punti doppi, cioè supponiamo che $P_1, P_2 \in \mathcal{C}^3$, $P_1 \neq P_2$ siano dei punti doppi. Indicata con P_1P_2 la retta passante per P_1 e P_2, si ha che

$$\left|P_1P_2 \cap \mathcal{C}^3\right| \geq 2+2=4,$$

pertanto P_1P_2 è una componente di $\mathcal{C}^3$ per il **Teorema di Bezout**, ma ciò contraddice l'irriducibilità di quest'ultima.

□

Quindi una cubica è

1. riducibile se si può esprimere come unione di tre rette o come unione di una retta e di una conica non degenere;
2. singolare se e solo se contiene un punto doppio.

Nel 1935, **Trygve Nagell**, pubblicò un procedimento per trasformare una cubica irriducibile, non singolare, avente un punto $\mathbb{K}$-razionale, nella **forma di Weierstrass**:

$$y^2 + y(a_1x + a_3) = x^3 + a_2x^2 + a_4x + a_6.$$

Figura 11.1: Trygve Nagell (1895-1988)

Teorema 11.10. *Sia $\mathcal{C}^3$ una cubica, nel piano affine $AG(2,\mathbb{K})$ si può effettuare un cambiamento di coordinate in modo tale che nel nuovo sistema di coordinate si verifichi:*

a) se $char(\mathbb{K}) \neq 2, 3$, la curva $\mathcal{C}^3$ ha equazione

$$y^2 = x^3 + Ax + B; \tag{11.1}$$

b) se $char(\mathbb{K}) = 3$, la curva $\mathcal{C}^3$ ha equazione

$$y^2 = x^3 + Ax^2 + Bx + C; \tag{11.2}$$

c) (i) se $char(\mathbb{K}) = 2$ e $a_1 = 0$, l'equazione è del tipo:

$$y^2 + Cy = x^3 + Ax + B \tag{11.3}$$

*ed in tal caso $\mathcal{C}^3$ si dice **supersingolare**;*

(ii) se $char(\mathbb{K}) = 2$ e $a_1 \neq 0$, l'equazione è del tipo:

$$y^2 + xy = x^3 + Ax^2 + B \tag{11.4}$$

*ed in tal caso $\mathcal{C}^3$ si dice **non supersingolare**.*

*Le equazioni del tipo (11.1) – (11.4) si dicono **equazioni di Weierstrass** di una cubica.*

Dimostrazione. Per il risultato di **Nagell**, una cubica (non singolare, irriducibile e con un punto $\mathbb{K}$-razionale) si può sempre scrivere nella forma

$$y^2 + y\,(a_1x + a_3) = x^3 + a_2x^2 + a_4x + a_6. \tag{11.5}$$

Supponiamo che $char\,(\mathbb{K}) \neq 2$. Allora, nell'equazione (11.5) si può effettuare il seguente cambiamento di coordinate:

$$\begin{cases} x &= X \\ y &= Y - \frac{a_1X+a_3}{2} \end{cases}$$

ottenendo:

$$\left(Y - \frac{a_1X + a_3}{2}\right)^2 + \left(Y - \frac{a_1X + a_3}{2}\right)(a_1X + a_3) = X^3 + a_2X^2 + a_4X + a_6.$$

Da tale equazione segue che

$$Y^2 - \frac{a_1^2X^2}{4} - \frac{a_3^2}{4} - \frac{a_1a_3X}{2} = X^3 + a_2X^2 + a_4X + a_6$$

e quindi l'equazione (11.5) nelle nuove coordinate è:

$$Y^2 = X^3 + \left(a_2 + \frac{a_1^2}{4}\right)X^2 + \left(a_4 + \frac{a_1a_3}{2}\right)X + \left(a_6 + \frac{a_3^2}{4}\right)$$

cioè essa è della forma:

$$Y^2 = X^3 + AX^2 + BX + C \tag{11.6}$$

con $A = \left(a_2 + \frac{a_1^2}{4}\right)$, $B = \left(a_4 + \frac{a_1a_3}{2}\right)$ e $C = \left(a_6 + \frac{a_3^2}{4}\right)$. Questo prova l'asserto *(b)*.

Inoltre, se la caratteristica del campo $\mathbb{K}$ è diversa anche da 3, si può considerare nell'equazione (11.6) un ulteriore cambiamento di coordinate effettuando la seguente sostituzione:

$$\begin{cases} X &= \tilde{X} - \frac{A}{3} \\ Y &= \tilde{Y} \end{cases}$$

Otteniamo così:

$$\tilde{Y}^2 = \left(\tilde{X} - \frac{A}{3}\right)^3 + A\left(\tilde{X} - \frac{A}{3}\right)^2 + B\left(\tilde{X} - \frac{A}{3}\right) + C$$

da cui ricaviamo

$$\tilde{Y}^2 = \tilde{X}^3 + \left(B - \frac{A^2}{3}\right)\tilde{X} + \left(\frac{2}{27}A^3 - \frac{AB}{3} + C\right).$$

Pertanto, l'equazione (11.6) nelle nuove coordinate, sarà della forma

$$\tilde{Y}^2 = \tilde{X}^3 + A'\tilde{X} + B'$$

con $A' = \left(B - \frac{A^2}{3}\right)$ e $B' = \left(\frac{2}{27}A^3 - \frac{AB}{3} + C\right)$, che è l'asserto *(a)*.

Infine supponiamo che $char\mathbb{K} = 2$, consideriamo l'equazione (11.5) e distinguiamo due casi.

Se $a_1 = 0$, effettuiamo il seguente cambiamento di coordinate:

$$\begin{cases} x &= X + a_2 \\ y &= Y \end{cases}$$

ottenendo così:

$$Y^2 + a_3 Y = (X + a_2)^3 + a_2 (X + a_2)^2 + a_4 (X + a_2) + a_6$$

da cui segue:

$$Y^2 + a_3 Y = X^3 + 3X^2 a_2 + 3X a_2^2 + a_2^3 + a_2 X^2 + 2a_2^2 X + a_2^3 + a_4 X + a_2 a_4 + a_6.$$

Ricordando che $char\mathbb{K} = 2$, otteniamo:

$$Y^2 + a_3 Y = X^3 + \left(a_2^2 + a_4\right) X + \left(a_2 a_4 + a_6\right).$$

Tale equazione è della forma

$$Y^2 + CY = X^3 + AX + B.$$

con $C = a_3$, $A = a_2^2 + a_4$, $B = a_2 a_4 + a_6$. Pertanto, nel caso $a_1 = 0$, si ha un'equazione del tipo (11.3).

Se $a_1 \neq 0$, consideriamo nell'equazione (11.5) la seguente trasformazione

$$\begin{cases} x &= a_1^2 X + a_1^{-1} a_3 \\ y &= a_1^3 Y + a_1^{-3}\left(a_1^2 a_4 + a_3^2\right) \end{cases}$$

Risulta:

$$\left(a_1^3 Y + a_1^{-1} a_4 + a_1^{-3} a_3^2\right)^2 + \left(a_1^3 Y + a_1^{-1} a_4 + a_1^{-3} a_3^2\right)\left(a_1\left(a_1^2 X + a_1^{-1} a_3\right) + a_3\right)$$

che a sua volta è uguale a

$$\left(a_1^2 X + a_1^{-1} a_3\right)^3 + a_2\left(a_1^2 X + a_1^{-1} a_3\right)^2 + a_4\left(a_1^2 X + a_1^{-1} a_3\right) + a_6,$$

da cui segue

$$\begin{aligned} a_1^6 Y^2 + a_1^6 XY =& a_1^6 X^3 + \left(a_1^3 a_3 + a_2 a_1^4\right) X^2 + \\ &+ \left(a_1^{-3} a_3^3 + a_2 a_1^{-2} a_3^2 + a_4 a_1^{-1} a_3 - a_1^{-2} a_4^2 - a_1^{-6} a_3^4 + a_6\right). \end{aligned}$$

Poichè $a_1 \neq 0$ possiamo dividere per a_1^6, ottenendo l'equazione

$$\begin{aligned} Y^2 + XY =& X^3 + \left(a_1^{-3} a_3 + a_2 a_1^{-2}\right) X^2 + \\ &+ \left(a_1^{-9} a_3^3 + a_2 a_1^{-8} a_3^2 + a_4 a_1^{-7} a_3 - a_1^{-8} a_4^2 - a_1^{-12} a_3^4 + a_6 a_1^{-6}\right) \end{aligned}$$

che è del tipo:

$$Y^2 + XY = X^3 + AX^2 + B$$

con coefficienti: $B = \left(a_1^{-9} a_3^3 + a_2 a_1^{-8} a_3^2 + a_4 a_1^{-7} a_3 - a_1^{-8} a_4^2 - a_1^{-12} a_3^4 + a_6 a_1^{-6}\right)$ e $A = \left(a_1^{-3} a_3 + a_2 a_1^{-2}\right)$ cioè un'equazione del tipo (11.4).

□

Per il **Teorema 11.10**, in seguito considereremo sempre cubiche rappresentate da un'equazione di Weierstrass, tenendo presente che il loro punto improprio, che denoteremo con ∞, ha coordinate proiettive omogenee $(0, 1, 0)$.

Poiché è facile vedere che il punto ∞ non è mai doppio per una cubica rappresentata da un'equazione di Weierstrass, dalla **Proposizione 11.6** segue che l'eventuale punto doppio di una cubica in forma di Weierstrass è proprio e quindi ha coordinate (x_0, y_0) tali che $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$.

Definizione 11.11. **(Curva Ellittica)**
Una cubica non singolare di $AG(2, \mathbb{K})$ rappresentata, a seconda della caratteristica del campo, da una delle equazioni di Weierstrass (11.1) − (11.4) del **Teorema 11.10**, si dice **curva ellittica**.

Teorema 11.12. *Valgono i seguenti risultati:*

1) *Se* $char(\mathbb{K}) \neq 2, 3$, *allora la cubica di equazione*
$$y^2 = x^3 + Ax + B$$
è singolare se e solo se vale $27B^2 + 4A^3 = 0$;

2) *Se* $char(\mathbb{K}) = 3$, *allora la cubica di equazione*
$$y^2 = x^3 + Ax^2 + Bx + C$$
è singolare se e solo se vale $-A^3C + A^2B^2 - B^3 = 0$;

3) *Se* $char\mathbb{K} = 2$, *si verifica che:*

   (i) *la cubica di equazione*
   $$y^2 + Cy = x^3 + Ax + B$$
   è singolare se e solo se $C = 0$;

   (ii) *la cubica di equazione*
   $$y^2 + xy = x^3 + Ax^2 + B$$
   è singolare se e solo se $B = 0$.

Dimostrazione. Se $char(\mathbb{K}) \neq 2, 3$, $\mathcal{C}^3$ ha equazione
$$y^2 = x^3 + Ax + B.$$
Affinchè $\mathcal{C}^3$ sia singolare, si deve verificare che: $f_y = 2y = 0$ e $f_x = 3x^2 + A = 0$. Cioè la cubica è singolare se e solo se
$$x^3 + Ax + B = 0, \tag{11.7}$$
ed inoltre
$$x^2 = -\frac{A}{3}, \tag{11.8}$$
rispettivamente.

Sostituendo tale valore di x^2 in (11.7) si ha:

$$x\left(-\frac{A}{3}\right) + Ax + B = 0$$

da cui segue:

$$\frac{2}{3}Ax + B = 0. \tag{11.9}$$

Se $A \neq 0$, da (11.9) si ottiene $x = -\frac{3B}{2A}$. Sostituendo quest'ultima espressione in (11.8), otteniamo $\frac{9B^2}{4A^2} = -\frac{A}{3}$. Pertanto segue che, in questo caso, condizione necessaria e sufficiente affinchè $\mathcal{C}^3$ sia singolare, è che:

$$27B^2 + 4A^3 = 0. \tag{11.10}$$

Se $A = 0$, allora condizione affinchè $\mathcal{C}^3$ sia singolare, è che sia $B = 0$ e quindi anche in questo caso la condizione di singolarià della cubica può essere espressa ancora dalla (11.10).

Se $char\,(\mathbb{K}) = 3$, $\mathcal{C}^3$ ha equazione $y^2 = x^3 + Ax^2 + Bx + C$ e affinchè $\mathcal{C}^3$ sia singolare, si deve verificare che $f_y = 2y = 0$ e $f_x = 3x^2 + 2Ax + B = 0$. Dalla prima condizione segue che

$$x^3 + Ax^2 + Bx + C = 0 \tag{11.11}$$

mentre la derivata rispetto ad x diventa $2Ax + B = 0$.

Se $A \neq 0$, si ha che $x = -\frac{B}{2A}$ e sostituendo tale valore di x in (11.11), si ottiene

$$-\frac{B^3}{8A^3} + \frac{AB^2}{4A^2} - \frac{B^2}{2A} + C = 0,$$

da cui segue $\frac{-B^3+2A^2B^2-4A^2B^2+8A^3C}{8A^3} = 0$. Pertanto, in questo caso, poichè $char\mathbb{K} = 3$, condizione necessaria e sufficiente affinchè $\mathcal{C}^3$ sia singolare, è che:

$$-A^3C + A^2B^2 - B^3 = 0 \tag{11.12}$$

Se $A = 0$, allora condizione affinchè $\mathcal{C}^3$ sia singolare, è che sia $B = 0$ e quindi anche in questo caso la condizione di singolarità della cubica può essere ancora espressa dalla (11.12).

Se $char\mathbb{K} = 2$, bisogna distinguere due sottocasi:

(i) **$\mathcal{C}^3$ ha equazione $y^2 + Cy = x^3 + Ax + B$.**

Analogamente ai casi precedenti, si deve verificare che:

$$f_y = 2y + C = 0 \text{ e } f_x = 3x^2 + A = 0.$$

Da tali equazioni, segue rispettivamente: $C = 0$ e $x^2 + A = 0$ che equivale a $x^2 = A$. Sostituendo tali risultati nell'equazione della $\mathcal{C}^3$, si ottiene $y^2 = Ax + Ax + B$ da cui segue $y^2 = B$. Quindi il punto di coordinate $(x, y) = \left(\sqrt{A}, \sqrt{B}\right)$ è un punto doppio (si noti che in un campo a caratteristica 2, tutti gli elementi sono quadrati).

(ii) $\mathcal{C}^3$ **ha equazione** $y^2 + xy = x^3 + Ax^2 + B$**.**

In tal caso si deve verificare che

$$f_y = 2y + x = 0 \text{ e } f_x = 3x^2 + 2Ax - y = 0.$$

Da tali condizioni segue che $x = 0$ e $y = 0$. Sostituendo i valori di x e y appena trovati nell'equazione di $\mathcal{C}^3$ si ottiene che, in questo caso, condizione necessaria e sufficiente affinchè la cubica sia singolare, è che sia $B = 0$.

□

Osservazione 11.13. Osserviamo che il **Teorema 11.12** fornisce delle condizioni necessarie e sufficienti affinchè una cubica in forma di Weierstrass sia una curva ellittica.

Abbiamo visto che una cubica è singolare se e solo se contiene un punto doppio. In tal caso, il seguente Teorema descrive l'equazione della cubica e le equazioni delle tangenti alla curva nel punto doppio, nel caso che questo sia una cuspide, un nodo o un punto doppio isolato.

Teorema 11.14. *Supponiamo che* $\mathrm{char}\mathbb{K} \neq 2, 3$ *e consideriamo una generica* $\mathcal{C}^3$ *su* $\mathbb{K}$ *. Valgono i seguenti risultati:*

(i) Se $\mathcal{C}^3$ *ha una cuspide, essa ha coordinate* $(0,0)$*. Allora la cubica ha equazione* $y^2 = x^3$ *e l'equazione complessiva delle tangenti in* $(0,0)$ *è* $y^2 = 0$*.*

(ii) Se $\mathcal{C}^3$ *ha un nodo o un punto doppio isolato allora, dopo un'opportuna trasformazione affine, esso ha coordinate* $(0,0)$*, la cubica ha equazione* $y^2 = x^2(x + a)$ $(a \neq 0)$ *e le tangenti nel suo punto doppio hanno equazione* $y - \alpha x = 0$ *e* $y + \alpha x = 0$*, dove* $\alpha^2 = a$ *e* $\alpha \in \mathbb{K}$ *nel caso del nodo o* $\alpha \notin \mathbb{K}$ *nel caso del punto doppio isolato.*

Dimostrazione. Dalla dimostrazione del **Teorema 11.12** segue che l'eventuale punto doppio di una cubica di equazione $y^2 = x^3 + Ax + B$ ha come prima coordinata una radice multipla del trinomio $x^3 + Ax + B$. Quindi si possono verificare due possibilità: $x^3 + Ax + B$ ha una radice tripla o doppia.

- Supponiamo che $x^3 + Ax + B$ abbia una radice tripla x_0. Allora x_0 è radice doppia e semplice delle derivate formali prime e seconde di $y^2 = x^3 + Ax + B$ rispettivamente. Poiché la derivata formale seconda è $6x$, allora $x_0 = 0$. Quindi $A = B = 0$ e pertanto la curva ha equazione

$$y^2 = x^3. \tag{11.13}$$

e il suo unico punto singolare ha coordinate $(0,0)$. Per la **Proposizione 11.6** l'equazione complessiva delle tangenti principali in $(0,0)$ ad $\mathcal{E}$, è data da $y^2 = 0$. Pertanto $(0,0)$ è una cuspide per (11.13).

- Supponiamo ora che $x^3 + Ax + B$ abbia una radice doppia x_0. Allora x_0 è radice semplice della derivata formale prima $3x^2 + A$. Pertanto si ha che:

$$x_0^3 + Ax_0 + B = 0 \tag{11.14}$$

ed inoltre

$$3x_0^2 + A = 0, \tag{11.15}$$

rispettivamente. Da (11.15) segue che $x_0^2 = -\frac{A}{3}$. Sostituendo tale valore in (11.14) si ottiene $x_0\left(-\frac{A}{3}\right) + Ax_0 + B = 0$ da cui segue che, se $A \neq 0$, $x_0 = -\frac{3B}{2A}$. Pertanto, l'unico punto singolare della cubica ha coordinate $(x_0, y_0) = \left(-\frac{3B}{2A}, 0\right)$. Se consideriamo la trasformazione affine:

$$\begin{cases} X &= x + \frac{3B}{2A} \\ Y &= y, \end{cases} \tag{11.16}$$

possiamo assumere che anche in questo caso l'unico punto singolare della curva sia il punto di coordinate $(0,0)$. Per trovare l'equazione della cubica, osserviamo che, essendo x_0 una radice doppia di $x^3 + Ax + B$, possiamo scrivere

$$y^2 = \left(x + \frac{3B}{2A}\right)^2 (x - x_1)$$

dove x_1 è una radice semplice del trinomio che descrive la curva. Effettuando il cambiamento di coordinate (11.16), si ha che

$$Y^2 = \left(X - \frac{3B}{2A} + \frac{3B}{2A}\right)^2 \left(X - \frac{3B}{2A} - x_1\right) = X^2 (X + a)$$

dove $a = -\frac{3B}{2A} - x_1$. Pertanto, nel caso in cui il trinomio abbia una radice doppia x_0, la curva ha un'equazione del tipo :

$$y^2 = x^2 (x + a) \tag{11.17}$$

per un opportuno $a \neq 0$. Per la **Proposizione 11.6** l'equazione complessiva delle tangenti principali in $(0,0)$ ad $\mathcal{E}$, è $y^2 - ax^2 = 0$, cioè le due tangenti nel punto doppio sono:

$$y - \alpha x = 0 \text{ e } y + \alpha x = 0.$$

Quindi nell'unico punto doppio di $\mathcal{E}$ ci sono due tangenti distinte. Pertanto si possono verificare due casi: $\alpha \in \mathbb{K}$ e l'origine è un nodo, o $\alpha \notin \mathbb{K}$ e l'origine è un punto doppio isolato.

□

11.2 Legge di gruppo

L'insieme dei punti di una curva ellittica $\mathcal{E}$, gode di un'importante proprietà: quella di avere una struttura di gruppo abeliano rispetto all'operazione di "somma" tra punti. Vediamo cosa significa sommare punti su una curva ellittica considerando dapprima il caso di una cubica **non singolare**.

Supponiamo innanzitutto che $char\mathbb{K} \neq 2, 3$, pertanto, considerata una generica curva ellittica $\mathcal{E}$ su $\mathbb{K}$, essa ha equazione $y^2 = x^3 + Ax + B$. Siano $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$ due punti non necessariamente distinti di $\mathcal{E}$. Se consideriamo la retta P_1P_2, per il **Teorema di Bezout**, essa interseca $\mathcal{E}$ in esattamente tre punti. Indicato con $P_3' = \left(x_{P_3'}, y_{P_3'}\right)$ il terzo punto di intersezione tra P_1P_2 ed $\mathcal{E}$, denotiamo con $P_3 = (x_3, y_3)$ il suo simmetrico rispetto all'asse x e poniamo:

$$P_1 + P_2 := P_3.$$

Questa costruzione vale sia se $P_1 \neq P_2$, sia se $P_1 = P_2$. In particolare si possono presentare quattro diversi casi:

- **Caso 1:** $P_1 \neq P_2$ **e** $P_1, P_2 \neq \infty$**.**

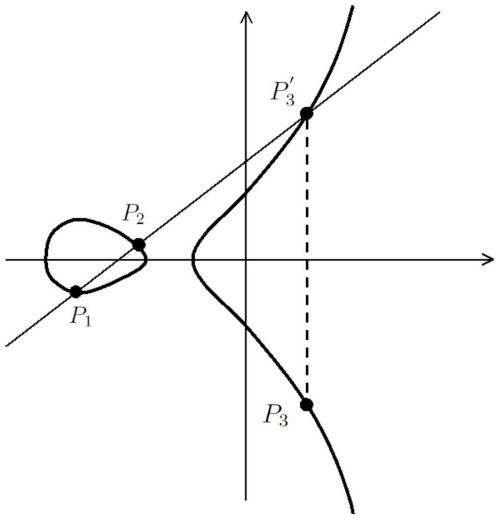


Figura 11.2: Somma tra punti di una curva ellittica nel **caso 1** con $x_1 \neq x_2$

Se $x_1 \neq x_2$, la retta P_1P_2 ha equazione:

$$y = m(x - x_1) + y_1. \tag{11.18}$$

con $m = \frac{y_2 - y_1}{x_2 - x_1}$. Per trovare le intersezioni di P_1P_2 con $\mathcal{E}$, sostituiamo (11.18) nell'equazione di $\mathcal{E}$ ed otteniamo

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B \tag{11.19}$$

che può essere riscritto nella forma:

$$x^3 - m^2x^2 + (A + 2mx_1 - 2my_1)x + \left(B - m^2x_1^2 + 2my_1x_1 - y_1^2\right) = 0 \tag{11.20}$$

Le tre radici di questa cubica sono le ascisse dei tre punti di intersezione di P_1P_2 con $\mathcal{E}$. Poichè x_1, x_2 sono radici di (11.20) e sono elementi di $\mathbb{K}$ e poichè il polinomio al primo membro in (11.20) è a coefficienti in $\mathbb{K}$, anche la terza radice $x_3 \in \mathbb{K}$. Quindi il polinomio al primo membro in (11.20) si fattorizza in

$$\begin{aligned}(x - x_1)(x - x_2)(x - x_3) &= x^3 - (x_1 + x_2 + x_3)x^2 + \\ &\quad + (x_2x_3 + x_1x_3 + x_1x_2)x - x_1x_2x_3.\end{aligned}$$

Pertanto, da (11.20) segue che $x_3 = m^2 - x_1 - x_2$ e quindi le coordinate del punto P_3 sono

$$\begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3). \end{cases} \tag{11.21}$$

Se invece $x_1 = x_2$, la retta P_1P_2 interseca $\mathcal{E}$ in ∞. Pertanto, in questo caso, si ha che $P_3 = \infty$.

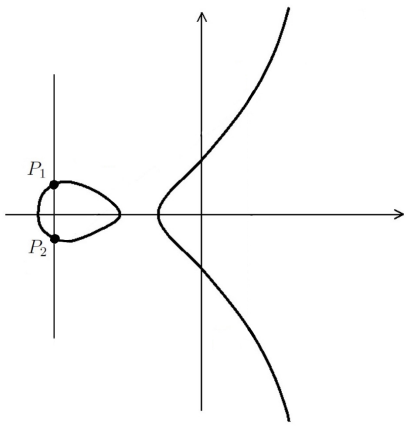


Figura 11.3: Somma tra punti di una curva ellittica nel **caso 1** con $x_1 = x_2$

- **Caso 2:** $P_1 = P_2$ **e** $P_1 \neq \infty$**.**

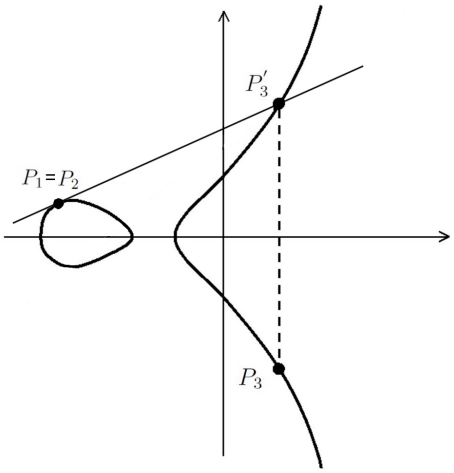


Figura 11.4: Somma tra punti di una curva ellittica nel **caso 2**

In questo caso, per la **Proposizione 11.6**, la tangente t in P_1 ad $\mathcal{E}$ ha equazione

$$f_x(x_1, y_1)(x - x_1) + f_y(x_1, y_1)(y - y_1) = 0,$$

cioè:

$$\left(-3x_1^2 - A\right)(x - x_1) + 2y_1(y - y_1) = 0.$$

Se $y_1 \neq 0$, allora l'equazione della tangente diventa la (11.18) dove $m = \frac{3x_1^2 + A}{2y_1}$. Ora, ragionando in modo analogo al caso precedente e tenendo presente che $x_1 = x_2$, si ha che, in questo caso, le coordinate di P_3 sono:

$$\begin{cases} x_3 = \left(\frac{3x_1^2 + A}{2y_1}\right)^2 - 2x_1 \\ y_3 = -y_1 + \left(\frac{3x_1^2 + A}{2y_1}\right)(x_1 - x_3). \end{cases} \tag{11.22}$$

Notiamo che, se $y_1 = 0$, allora $2P_1 = \infty$.

- **Caso 3:** $P_1 \neq P_2$ **e** $P_1 = \infty$ **(rispettivamente** $P_2 = \infty$ **).**

In questo caso, l'intersezione tra la retta $x = x_2$ (rispettivamente $x = x_1$) e la curva $\mathcal{E}$ contiene P_1, P_2 e $Q = (x_2, -y_2)$ (rispettivamente $Q = (x_1, -y_1)$). Pertanto si definisce $P_1 + P_2 = P_2$ (rispettivamente $P_1 + P_2 = P_1$).

- **Caso 4:** $P_1 = P_2 = \infty$**.**

In tal caso, per definizione, si pone $\infty + \infty = \infty$.

La costruzione sopra descritta vale anche nei casi in cui $char\mathbb{K} = 2$ e $char\mathbb{K} = 3$. Tuttavia, in tali casi, a differenza del caso precedente, le coordinate di P_3 si ricavano da quelle di $P_3^{'}$ attraverso la relazione[1]:

$$(x_3, y_3) = \left(x_{P_3'}, -a_1 x_{P_3'} - a_3 - y_{P_3'}\right).$$

Inoltre, sia nel caso in cui $char\mathbb{K} = 2$, che nel caso in cui $char\mathbb{K} = 3$, continuano a valere i risultati riportati nel terzo e nel quarto caso della costruzione descritta sopra. Per i primi due casi, invece, si ottengono risultati diversi in base alla caratteristica del campo, come riportato di seguito.

Se $char\mathbb{K} = 2$ e l'equazione della $\mathcal{C}^3$ è del tipo $y^2 + Cy = x^3 + Ax + B$, allora distinguiamo due casi:

Caso 1: $P_1 \neq P_2$ e $P_1, P_2 \neq \infty$.

- Se $x_1 \neq x_2$, le coordinate di P_3 sono:

$$\begin{cases} x_3 &= \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 + x_1 + x_2 \\ y_3 &= C + y_1 + \left(\frac{y_2-y_1}{x_2-x_1}\right)(x_3 + x_1) \end{cases}$$

- Se $x_1 = x_2$, allora $P_3 = \infty$.

Caso 2: $P_1 = P_2$ e $P_1 \neq \infty$.

- Se $C \neq 0$, le coordinate di $P_3 = 2P_1$ sono:

$$\begin{cases} x_3 &= \frac{x_1^4 + A^2}{C^2} \\ y_3 &= C + y_1 + \left(\frac{x_1^2 + A}{C}\right)(x_3 + x_1). \end{cases}$$

- Se $C = 0$, allora $2P_1 = \infty$.

Se invece $char\mathbb{K} = 2$ e l'equazione della $\mathcal{C}^3$ è del tipo $y^2 + xy = x^3 + Ax^2 + B$, allora si ha:

Caso 1: $P_1 \neq P_2$ e $P_1, P_2 \neq \infty$.

- Se $x_1 \neq x_2$, le coordinate di P_3 sono:

$$\begin{cases} x_3 &= A + \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 + \frac{y_2-y_1}{x_2-x_1} + x_1 + x_2 \\ y_3 &= y_1 + \left(1 + \frac{y_2-y_1}{x_2-x_1}\right)x_3 + \frac{y_2-y_1}{x_2-x_1}x_1. \end{cases}$$

- Se $x_1 = x_2$, allora $P_3 = \infty$.

[1] I coefficienti che compaiono nel calcolo delle coordinate di P_3 fanno ovviamente riferimento all'equazione $y^2 + y(a_1 x + a_3) - x^3 - a_2 x^2 - a_4 x - a_6 = 0$ in coordinate non omogenee di una curva ellittica.

Caso 2: $P_1 = P_2$ e $P_1 \neq \infty$.

- Se $x_1 \neq 0$, le coordinate di $P_3 = 2P_1$ sono:

$$\begin{cases} x_3 &= \frac{x_1^4+B}{x_1^2} \\ y_3 &= x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3. \end{cases}$$

- Se $x_1 = 0$, allora $2P_1 = \infty$.

Se $char\mathbb{K} = 3$ e quindi l'equazione della $\mathcal{C}^3$ è del tipo $y^2 = x^3 + Ax^2 + Bx + C$, allora si ha:

Caso 1: $P_1 \neq P_2$ e $P_1, P_2 \neq \infty$.

- Se $x_1 \neq x_2$, le coordinate di P_3 sono:

$$\begin{cases} x_3 &= \left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - A - x_1 - x_2 \\ y_3 &= -y_1 + \left(\frac{y_2-y_1}{x_2-x_1}\right)(x_1 - x_3). \end{cases}$$

- Se $x_1 = x_2$, allora $P_3 = \infty$.

Caso 2: $P_1 = P_2$ e $P_1 \neq \infty$.

- Se $y_1 \neq 0$, le coordinate di $P_3 = 2P_1$ sono:

$$\begin{cases} x_3 &= \left(\frac{2Ax_1+B}{2y_1}\right)^2 - A - 2x_1 \\ y_3 &= -y_1 + \left(\frac{2Ax_1+B}{2y_1}\right)(x_1 - x_3). \end{cases}$$

- Se $y_1 = 0$, allora $2P_1 = \infty$.

Vale il seguente Teorema.

Teorema 11.15. *La somma dei punti su una curva ellittica $\mathcal{E}$ soddisfa le seguenti proprietà:*

1. ***Proprietà associativa**: $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ per ogni $P_1, P_2, P_3 \in \mathcal{E}$.*
2. ***Esistenza dell'elemento neutro**: $P + \infty = P$ per ogni $P \in \mathcal{E}$.*
3. ***Esistenza dell'opposto**: Dato un punto $P = (x, y)$ su $\mathcal{E}$, allora il punto $P' = (x, -y)$ è l'unico punto di $\mathcal{E}$ tale che $P + P' = \infty$. Il punto P' si indica con $-P$.*
4. ***Proprietà commutativa**: $P_1 + P_2 = P_2 + P_1$ per ogni $P_1, P_2 \in \mathcal{E}$*

Quindi i punti di $\mathcal{E}$ costituiscono un gruppo abeliano rispetto alla somma e l'elemento neutro di tale gruppo è ∞.

Osservazione 11.16. Osserviamo che l'abelianità del gruppo segue banalmente dal fatto che $P_1P_2 = P_2P_1$.

Esempio 11.17. Ad esempio, la cubica a coefficienti in $GF(11)$

$$\mathcal{E} : y^2 = x^3 + x + 6$$

è una curva ellittica poiché $\left(4 \times 1^3 + 27 \times 6^2\right) \ mod\ 11 = 8$. Per determinare l'ordine di $\mathcal{E}$ è sufficiente verificare se $z = x^3 + x + 6$ è un quadrato al variare di x in $GF(11)$. Per il **criterio di Eulero**, z è un quadrato in $GF(11)$ se, e solo se, $z^{\frac{11-1}{2}} \equiv 1 \ mod\ 11$ e quando ciò si verifica è noto che le radici quadrate modulo 11 sono $\pm z^{\frac{11+1}{4}}$.

Numero punti di $\mathcal{E}$

x	$\left(x^3 + x + 6\right) \ mod\ 11$	Quadrato	y
0	6	no	
1	8	no	
2	5	si	$4, 7$
3	3	si	$5, 6$
4	8	no	
5	4	si	$2, 9$
6	8	no	
7	4	si	$2, 9$
8	9	si	$3, 8$
9	7	no	
10	4	si	$2, 9$

Dalla tabella si evince che l'ordine di $\mathcal{E}$ è 13. Quindi, $\mathcal{E} \cong Z_{13}$ e pertanto ogni punto affine di $\mathcal{E}$ è un generatore del gruppo ad essa associato. In particolare $P = (2, 7)$ è un generatore di $\mathcal{E}$.

Calcoliamo $2P = P + P = (2, 7) + (2, 7)$

$$2P = \begin{cases} x_3 = \left(\dfrac{3x_1 + a}{2y_1}\right)^2 - x_1 - x_2 = \\ \quad = \left(\left((3 \times 2^2 + 1)(2 \times 7)^{-1}\right)^2 - 2 - 7\right) \ mod\ 11 = 5 \\ y_3 = \left(\dfrac{3x_1 + a}{2y_1}\right)(x_1 - x_3) - y_1 \\ \quad = \left(\left((3 \times 2^2 + 1)(2 \times 7)^{-1} \times (2 - 8)\right) - 2\right) \ mod\ 11 = 2 \end{cases}$$

Quindi $2P = (5, 2)$.

Calcoliamo $3P = 2P + P$

$$3P = \begin{cases} x_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 = \\ \quad = \left(\left((7-2)(2-5)^{-1}\right)^2 - 5 - 2\right) \ mod\ 11 = 8 \\ y_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 = \\ \quad = \left(\left((7-2)(2-5)^{-1} \times (5-8)\right) - 2\right) \ mod\ 11 = 3 \end{cases}$$

Pertanto, $3P = (8, 3)$.

Procedendo in questo modo, si ottiene

$$\begin{array}{lll} P = (2,7) & 2P = (5,2) & 3P = (8,3) \\ 4P = (10,2) & 5P = (3,6) & 6P = (7,9) \\ 7P = (7,2) & 8P = (3,5) & 9P = (10,9) \\ 10P = (8,8) & 11P = (5,9) & 12P = (2,4) \end{array}$$

e chiaramente $13P = \infty$.

□

Se $\mathbb{K}$ è finito, la struttura generale del gruppo associato ad una curva ellittica è ben nota, come mostra il seguente teorema.

Teorema 11.18. *Sia $\mathcal{E}$ una curva ellittica sul campo finito $GF(q)$. Allora*

$$\mathcal{E} \cong Z_n \quad \textit{oppure} \quad \mathcal{E} \cong Z_{n_1} \oplus Z_{n_2}$$

dove n, n_1, n_2 sono opportuni interi ≥ 1 e $n_1 \mid n_2$.

11.3 Numero di punti di una curva ellittica

In diverse applicazioni in crittografia vengono utilizzate le curve ellittiche. Tali curve, se sono a coefficienti in un campo finito, hanno un numero finito di punti. Tale numero si indica con N ed è detto **ordine della curva $\mathcal{E}$**.

Un'altra grandezza importante nell'ambito della crittografia su curve ellittiche, è l'**ordine di un punto** della curva. Data una curva ellittica $\mathcal{E}$ ed un punto $P \in \mathcal{E}$, si dice ordine di P il più piccolo intero positivo k, se esiste, tale che $kP = \infty$. Se tale intero non esiste, si dice che P ha ordine infinito.

Vediamo ora di dare una stima del numero dei punti di una curva ellittica $\mathcal{E}$ definita su $GF(q)$ con $q = p^r$, p primo. Supponiamo che $\mathcal{E}$ abbia equazione

$$y^2 = x^3 + ax + b. \tag{11.23}$$

Abbiamo visto che se $p = 2$, $\mathcal{E}$ è data dalle equazioni

$$y^2 + cy = x^3 + ax + b \quad \text{o} \quad y^2 + xy = x^3 + ax + b \tag{11.24}$$

mentre se $p = 3$, $\mathcal{E}$ è definita da

$$y^2 = x^3 + ax^2 + bx + c. \tag{11.25}$$

E' facile vedere che una curva ellittica ha al più $2q + 1$ punti di $GF(q)$, cioè ∞ insieme con $2q$ coppie (x, y) con x, y elementi di $GF(q)$ che soddisfano (11.23) (o (11.24) o (11.25) se $p = 2$ o 3, rispettivamente). Infatti, per ciascuno dei q possibili valori di x ci sono al più 2 valori di y che soddisfano (11.23).

Poiché solo la metà degli elementi di $GF(q)^*$ hanno radici quadrate ci si aspetta che il numero N dei punti di una curva ellittica sia circa il numero dei punti di $GF(q)$. Più precisamente, sia χ il carattere quadratico di $GF(q)$. Cioè χ è la funzione che a $x \in GF(q)^*$ associa ± 1 a seconda che x abbia o meno una radice quadrata in $GF(q)$ (e prendiamo $\chi(0) = 0$). Per esempio, se $q = p$ è un primo, allora $\chi(x) = \left(\frac{x}{p}\right)$ è il simbolo di Legendre. Pertanto, in tutti i casi, il numero delle soluzioni $y \in GF(q)$ dell'equazione $y^2 = u$ è uguale a $1 + \chi(u)$ e quindi il numero delle soluzioni di (11.23) (contando anche il punto ∞) è

$$1 + \sum_{x \in GF(q)} \left(1 + \chi\left(x^3 + ax + b\right)\right) = q + 1 + \sum_{x \in GF(q)} \chi\left(x^3 + ax + b\right). \tag{11.26}$$

Ci si aspetta che $\chi\left(x^3 + ax + b\right)$ possa assumere con la stessa frequenza i valori $+1$ e -1. Il seguente Teorema fornisce dei limiti più precisi per il numero dei punti di $\mathcal{E}$:

Teorema 11.19. ***(Teorema di Hasse)***
Sia $\mathcal{E}$ una curva ellittica sul campo finito $GF(q)$. Allora l'ordine N di $\mathcal{E}$ soddisfa

$$|N - (q + 1)| \leq 2\sqrt{q}. \tag{11.27}$$

Osservazione 11.20. Osserviamo che (11.27) equivale a

$$\left(\sqrt{q} - 1\right)^2 \leq N \leq \left(\sqrt{q} + 1\right)^2 \tag{11.28}$$

Esempio 11.21. Si consideri la curva ellittica $\mathcal{E} : y^2 = x^3 + x + 6$ a coefficienti in $GF(11)$, vale che

$$5,36 \sim \left(\sqrt{11} - 1\right)^2 \leq N = 13 \leq \left(\sqrt{11} + 1\right)^2 \sim 18,63.$$

□

12 Crittosistemi basati su curve ellittiche

12.1 Costruzione di una curva ellittica

Nella costruzione di sistemi crittografici basati sulle curve ellittiche, la prima operazione da fare è la scelta di una curva ellittica $\mathcal{E}$ definita su $GF(q)$ e di un suo punto P. Entrambi possono essere determinati contemporaneamente dal seguente metodo probabilistico basato sul **Teorema 11.12**:

1. Si fissa un campo finito $GF(q)$;
2. A seconda della caratteristica del campo, si presentano situazioni diverse:

   2a) Supponiamo che la caratteristica del campo sia maggiore strettamente di 3. Quindi, in tal caso, si sceglie una terna (x_0, y_0, a) di elementi di $GF(q)$ e si calcola $b = y_0^2 - (x_0^3 + ax_0)$. Successivamente si calcola $4a^3 + 27b^2$.

   - Se $4a^3 + 27b^2 \neq 0$ in $GF(q)$, allora $\mathcal{E}: \ y^2 - x^3 - ax - b = 0$ è una curva ellittica e $P = (x_0, y_0)$ è un suo punto;
   - Se $4a^3 + 27b^2 = 0$ si ritorna a scegliere una terna (x_0', y_0', a') e si ripete il procedimento;

   2b) Se la caratteristica del campo è uguale a 3, si sceglie una terna (x_0, y_0, a) di elementi di $GF(q)$ e si calcola $c = y_0^2 - (x_0^3 + ax_0^2 + bx_0)$. Successivamente si calcola $-a^3c + a^2b^2 - b^3$.

   - Se $-a^3c + a^2b^2 - b^3 \neq 0$ in $GF(q)$, allora risulta che $\mathcal{E}: \ y^2 - x^3 - ax^2 - bx - c = 0$ è una curva ellittica e $P = (x_0, y_0)$ è un suo punto;
   - Se $-a^3c + a^2b^2 - b^3 = 0$ si ritorna a scegliere una terna (x_0', y_0', a') e si ripete il procedimento.

   2c) Se la caratteristica del campo è uguale a 2, si distinguono due sottocasi a seconda che la curva sia supersingolare o meno.

   - Se la curva che si vuole costruire è supersingolare, si sceglie una terna (x_0, y_0, a) di elementi di $GF(q)$ e si calcola $b = y_0^2 + cy_0 - (x_0^3 + ax_0)$. Successivamente si valuta c.
     - Se $c \neq 0$ in $GF(q)$, allora risulta che $\mathcal{E}: \ y^2 + cy - x^3 - ax^2 - b = 0$ è una curva ellittica e $P = (x_0, y_0)$ è un suo punto;
     - Se $c = 0$ si ritorna a scegliere una terna (x_0', y_0', a') e si ripete il procedimento.

- Se la curva che si vuole costruire è non supersingolare, si sceglie una terna (x_0, y_0, a) di elementi di $GF(q)$ e si calcola $b = y_0^2 + x_0y_0 - (x_0^3 + ax_0)$. Successivamente si valuta b.
  - Se $b \neq 0$ in $GF(q)$, allora risulta che $\mathcal{E}: y^2+xy-x^3-ax^2-b = 0$ è una curva ellittica e $P = (x_0, y_0)$ è un suo punto;
  - Se $b = 0$ si ritorna a scegliere una terna (x_0', y_0', a') e si ripete il procedimento.

12.2 Conversione delle unità di messaggio in chiaro in punti di una curva ellittica

Ora bisogna convertire le unità di messaggio in chiaro in punti di una fissata curva ellittica definita su un campo finito $GF(q)$, q dispari.

Fissata una curva ellittica $\mathcal{E}$ definita dall'equazione di Weierstrass $y^2 = f(x)$ su $GF(q)$ dove $q = p^r$ è un numero dispari elevato, il seguente metodo probabilistico permette di trasformare le unità di messaggio in chiaro in punti di $\mathcal{E}$.

Sia k un intero sufficientemente grande e supponiamo che

1. le unità di messaggio in chiaro siano rappresentate da interi m tali che $0 \leq m < M$,
2. $q = p^r$ sia scelto in modo tale che $q > Mk$.

Gli interi compresi tra 1 ed Mk si possono rappresentare nella forma $mk+j$, dove $1 \leq j \leq k$. Rappresentato in base p, l'intero $mk + j = \sum_{i=0}^{r-1} a_i p^i$ con $0 \leq a_i \leq p-1$. Quindi $mk + j$ corrisponde alla r–upla $(a_{r-1}, \ldots, a_1, a_0)$ dello spazio vettoriale $GF(p)^r$ e quindi ad un unico elemento $x_j \in GF(q)$. Pertanto, fissato m, per ogni $j = 1, 2, \ldots, k$ ad ogni intero $mk + j$ corrisponde un unico elemento x_j di $GF(q)$. Se $f(x_j)$ è un quadrato in $GF(q)$ e y_j è tale che $y_j^2 = f(x_j)$, allora la coppia (x_j, y_j) definisce il punto $P_m = (x_j, y_j)$ della curva ellittica $\mathcal{E}$ di equazione $y^2 = f(x)$. Se $f(x_j)$ non è un quadrato, allora si ripete il procedimento appena visto per l'elemento x_{j+1} corrispondente all'intero $mk + j + 1$.

Vediamo ora il procedimento inverso, cioè supponiamo di avere a disposizione un punto $Q = P_m$ della curva ellittica e di voler determinare l'unità di messaggio in chiaro m da cui esso proviene. Sia $Q \in \mathcal{E}$ un punto di coordinate (x_Q, y_Q). Poiché sussiste l'isomorfismo di spazi vettoriali $GF(q) \cong GF(p)^r$, x_Q è in corrispondenza biunivoca con la r–upla $(a_0, \ldots, a_{r-1})_p$ e quindi, per quanto visto in precedenza, anche con $\tilde{x}_Q = \sum_{i=0}^{r-1} a_i p^i$. Pertanto si ha che: $Q = P_m$ se e solo se $\tilde{x}_Q = mk + j$. Allora consideriamo $\tilde{x}_Q - 1 = mk + j - 1$ e dividiamo tutto per k ottenendo

$$\frac{\tilde{x}_Q - 1}{k} = \frac{mk + j - 1}{k} = m + \frac{j-1}{k}. \tag{12.1}$$

Osserviamo che $1 \leq j \leq k$, cioè $0 \leq j-1 \leq k-1$ da cui segue $0 \leq \frac{j-1}{k} < 1$. Pertanto, passando alla parte intera di (12.1), abbiamo:

$$\left\lfloor m + \frac{j-1}{k} \right\rfloor = m.$$

Quindi, se questo procedimento ha successo, esso permette di costruire una corrispondenza biunivoca tra l'insieme delle unità di messaggio in chiaro ed un opportuno sottoinsieme di punti di $\mathcal{E}$.

Vediamo quale è la probabilità che questo metodo per convertire un'unità di messaggio m in un punto P_m della curva ellittica $\mathcal{E}$, fallisca.

Consideriamo x_j e osserviamo che la probabilità che $f(x_j)$ sia un non quadrato (e quindi la probabilità che il metodo fallisca), è di circa $\frac{1}{2}$. Più precisamente, la probabilità che $f(x)$ sia un quadrato al variare di x in $GF(q)$, è data da $\frac{N}{2q}$, cioè dal rapporto tra il numero dei punti della curva ellittica $\mathcal{E}$ e il numero delle possibili coppie del tipo $(x, \pm\sqrt{f(x)})$ con $\sqrt{f(x)}$ in un'estensione quadratica di $GF(q)$. Da (11.28) segue che

$$\frac{\left(\sqrt{q}-1\right)^2}{2q} \leq \frac{N}{2q} \leq \frac{\left(\sqrt{q}+1\right)^2}{2q}$$

che può essere espresso nella forma:

$$\frac{1}{2}\left(1-\frac{1}{\sqrt{q}}\right)^2 \leq \frac{N}{2q} \leq \frac{1}{2}\left(1+\frac{1}{\sqrt{q}}\right)^2.$$

Quindi

$$\lim_{q\to+\infty} \frac{N}{2q} = \frac{1}{2}$$

Pertanto la probabilità che $f(x_j)$ sia un non quadrato per ogni $1 \leq j \leq k$, è circa $\frac{1}{2^k}$. Possiamo quindi osservare che per $k \to +\infty$, segue che $q \to +\infty$ siccome $q > Mk$, e quindi la probabilità di successo del metodo, che è data da $1 - \frac{1}{2^k}$, tende ad 1.

12.3 Logaritmo discreto su una curva ellittica

In seguito è fornita la definizione di logaritmo discreto nell'ambito del gruppo associato ad una curva ellittica.

Definizione 12.1. (Logaritmo discreto su una curva ellittica)
Sia $(\mathcal{E}, +)$ il gruppo associato ad una curva ellittica su $GF(q)$ e sia B un suo punto. Sia ora $P \in \langle B \rangle$ allora esiste $x \in \mathbb{Z}$ tale che $P = xB$. L'intero x si dice **logaritmo di P in base B** e lo si denota $\log_B P$.

Il seguente crittosistema è l'analogo del crittosistema di ElGamal e fonda la sua sicurezza sull'intrattabilità computazione del problema del logaritmo discreto su curve ellittiche.

Definizione 12.2. (Crittosistema di ElGamal basato sulle Curve Ellittiche)
Siano $\mathcal{E}$ una curva ellittica su $GF(q)$ e sia B un suo punto di ordine n tale che il PLD in $(\langle B \rangle, +)$ è computazionalmente intrattabile. Inoltre siano:

1. $\mathcal{P} = \mathcal{E}$;
2. $\mathcal{C} = \mathcal{E} \times \mathcal{E}$;
3. $\mathcal{K} = \{(\mathcal{E}, B, m, P, n) : mB = P\}$, dove $(\mathcal{E}, P, B, n)$ è pubblica, mentre m è segreta.
4. Per $K = (\mathcal{E}, B, m, P, n)$ ed un elemento random (segreto) $k \in \mathbb{Z}_n^*$ vale che

$$\begin{aligned} e_{K,k} &: \mathcal{E} \longrightarrow \mathcal{E} \times \mathcal{E}, X \longmapsto (kB, X + k(mB)) \\ d_K &: \mathcal{E} \times \mathcal{E} \longrightarrow \mathcal{E}, (Y_1, Y_2) \longmapsto Y_2 - mY_1. \end{aligned}$$

Esempio 12.3. Si consideri la curva ellittica $\mathcal{E} : y^2 = x^3 + x + 6$ a coefficienti in $GF(11)$. Abbiamo visto nell'**Esempio 11.17** che $\mathcal{E} \cong Z_{13}$ e $B = (2, 7)$ è un suo generatore. L'utente sceglie, per esempio, $m = 7$ come chiave segreta e quindi $P = 7B = (7, 2)$. Allora la chiave pubblica è $(\mathcal{E}, (7, 2), (2, 7), 13)$. Viene generato un intero random $k \in \mathbb{Z}_{13}^*$. Allora

$$\begin{aligned} e_{K,3} &: \mathcal{E} \longrightarrow \mathcal{E} \times \mathcal{E}, X \longmapsto (k(7, 2), X + k(2, 7)) \\ d_K &: \mathcal{E} \times \mathcal{E} \longrightarrow \mathcal{E}, (Y_1, Y_2) \longmapsto Y_2 - 7Y_1. \end{aligned}$$

Se $X = (10, 9)$ e $k = 3$, allora l'utente trasmette

$$e_{K,k}(10, 9) = (3(7, 2), (10, 9) + 3(2, 7)) = ((8, 3), (10, 2)).$$

Una volta ricevuto $((8,3),(10,2))$, il destinatario prestabilito calcola

$$X = (8,3) - 7(10,2) = (8,3) - (3,5) = (8,3) + (3,6) = (10,9)$$

che è il messaggio originario.

□

Un crittosistema di tipo ElGamal è il crittosistema integrato basato sulle curve ellittiche (ECIES). E' qui di seguito presentata una versione semplificata.

Sia $\mathcal{E} : y^2 = x^3 + ax + b$ una curva ellittica su $GF(p)$, p primo, $p > 2$, e sia $P = (x,y)$ un suo punto proprio. Poiché

$$y \ mod\ p + (-y) \ mod\ p = p$$

e p è dispari allora $y \ mod\ p$ e $(-y) \ mod\ p$ hanno classe di parità distinta. Quindi, P è completamente determinato dalla coppia $(x, y \ mod\ 2)$ con $0 \leq x, y \leq p-1$. Viene così definita la seguente applicazione:

$$\text{PointCompress} : \mathcal{E} - \{\infty\} \longrightarrow \mathbb{Z}_p \times \mathbb{Z}_2, (x,y) \longmapsto (x, y \ mod\ 2)$$

Viceversa, il recupero di P da $(x, y \ mod\ 2)$ si ottiene dal seguente algoritmo

Algoritmo 12.4. **(PointDecompress(x,i))**

$z \longleftarrow x^3 + ax + b$
if z non è un residuo quadratico modulo p
then return ("Insuccesso")
else
$y \longleftarrow \sqrt{z} \ mod\ p$
if $y \equiv i \ mod\ 2$
then return (x,y)
else return $(x, p-y)$

Per il **Criterio di Eulero**, z è un quadrato se, e solo se, $z^{\frac{p-1}{2}} \equiv 1 \ mod\ p$ e inoltre le radici sono $\pm z^{\frac{p+1}{4}}$ se $p \equiv 3 \ mod\ 4$.

Usando, l'applicazione PointCompress i bit da memorizzare corrispondenti ai punti della curva ellittica sono al più il 50% a costo di calcoli aggiuntivi per il recupero delle coordinata y del punto.

Definizione 12.5. **(ECIES Semplificato)**
Siano $\mathcal{E}$ una curva ellittica su $GF(p)$ e sia B un suo punto di ordine n tale che il PLD in $(\langle B\rangle, +)$ è computazionalmente intrattabile. Inoltre siano:

1. $\mathcal{P} = \mathbb{Z}_p$;
2. $\mathcal{C} = (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p^*$;
3. $\mathcal{K} = \{(\mathcal{E}, B, m, P, n) : mB = P\}$, dove $(\mathcal{E}, P, B, n)$ è pubblica, mentre m è segreta.
4. Per $K = (\mathcal{E}, B, m, P, n)$ ed un elemento random (segreto) $k \in \mathbb{Z}_n^*$ e $kP = (x_0, y_0)$ e $x_0 \neq 0$, allora

$$e_{K,k} : \mathbb{Z}_p \to (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p^*, x \mapsto (\text{PointCompress}(kB), xx_0 \ mod\ p)$$
$$d_K : (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p^* \to \mathbb{Z}_p, (y_1, y_2) \mapsto y_2 \left(x_0^{-1}\right) \ mod\ p.$$

dove $(x_0, y_0) = m\text{PointDecompress}(y_1)$.

Esempio 12.6. Si consideri la curva ellittica $\mathcal{E} : y^2 = x^3 + x + 6$ a coefficienti in $GF(11)$ dell'**Esempio 11.17**. Abbiamo visto che $\mathcal{E} \cong Z_{13}$ e $B = (2, 7)$ è un suo generatore.

L'utente sceglie, per esempio, $m = 7$ come chiave segreta e quindi $P = 7B = (7, 2)$. Supponiamo che voglia cifrare il testo in chiaro $x = 9$ e che abbia scelto il numero casuale $k = 6$. Allora si calcolano

$$\begin{aligned} 6B &= 6(2,7) = (7,9) \\ 6P &= 6(7,2) = (8,3) \end{aligned}$$

Quindi $x_0 = 8$. Successivamente, si calcola

$$e_{K,6}(9) = (\text{PointCompress}(7,9), (9 \times 8) \ mod\ 11) = ((7,1),6)$$

Una volta ricevuto $((7,1),6)$, il destinatario calcola

$$\begin{aligned} \text{PointDecompress}(7,1) &= (7,9) \\ 7(7,9) &= (8,3) \end{aligned}$$

e quindi

$$d_{K,6}(((7,1),6)) = 6 \times 8^{-1} \ mod\ 11 = 9$$

che era il testo in chiaro originario.

□

12.4 Fattorizzare attraverso l'uso delle curve ellittiche

In questa sezione tratteremo un'ulteriore applicazione delle curve ellittiche che riguarda la fattorizzazione di un intero composito dispari. Tale applicazione è nota come **algoritmo di Lenstra**.

Prima di procedere alla descrizione di questo metodo, premettiamo alcuni risultati relativi alle rappresentazioni binarie con segno che permettono di calcolare più speditamente i multipli di un punto di una curva ellittica.

12.4.1 Rappresentazione NAF

Definizione 12.7. **(Rappresentazione binaria con segno)**
Sia c un intero, una **rappresentazione binaria con segno** di c è una ℓ-pla $(c_{\ell-1}, ..., c_0)$ con $c_i \in \{-1, 0, 1\}$ per ogni $0 \leq i \leq \ell - 1$ tale che

$$c = \sum_{i=0}^{\ell-1} c_i 2^i.$$

Esempio 12.8. In generale esiste più di una rappresentazione binaria di uno stesso intero. Infatti,

$$11 = 2^3 + 2^1 + 2^0 = 2^4 - 2^2 - 2^0$$

e quindi $(0, 1, 0, 1, 1)$ e $(1, 0, -1, 0, -1)$ sono rappresentazioni binarie con segno di 11.

□

Definizione 12.9. **(Rappresentazione NAF)**
Una rappresentazione binaria con segno $(c_{\ell-1}, ..., c_0)$ di un intero c tale che $(c_{i+1}, c_i) \neq (\pm 1, \pm 1)$ si dice **forma non adiacente** (o, semplicemente, rappresentazione **NAF**).

Teorema 12.10. *Ogni intero ammette un'unica rappresentazione NAF.*

Dimostrazione.

- **Esistenza.** Sia $(b_{k-1}, ..., b_0)$ la rappresentazione binaria del generico intero c. Quindi,
$$c = \sum_{j=0}^{k-1} b_j 2^j,\ b_j \in \{0,1\}.$$
Sia
$$h = \min\{0 \le j \le k-1 : (b_{j+1}, b_j) = (1,1)\},$$
allora $b_{h-1} = 0$ e quindi
$$c = \sum_{j=h+2}^{k-1} b_j 2^j + 2^{h+1} + 2^h + \sum_{j=0}^{h-1} b_j 2^j =$$
$$= \sum_{j=h+2}^{k-1} b_j 2^j + 2^{h+2} - 2^h + \sum_{j=0}^{h-1} b_j 2^j.$$
Quindi,
$$c = \sum_{j=0}^{k-1} b'_j 2^j$$
dove $b'_j = b_j$ per $0 \le j \le h-1$ con $b'_{h-1} = b_{h-1} = 0$, e dove $b_h = -1$ e $b_{h+1} = 0$. Quindi,
$$h + 2 \le \min\left\{0 \le j \le k-1 : (b'_{j+1}, b'_j) = (1,1)\right\}.$$
Pertanto dopo al piu $k - 1 - (h-2) + 1 = k - h + 2$ passi si ottiene una rappresentazione in NAF.

- **Unicità.** Supponiamo che
$$c = \sum_{i=0}^{m-1} c_i 2^i = \sum_{j=0}^{n-1} c'_j 2^j$$
siano due rappresentazioni NAF distinte dell'intero c. Sia
$$e = \max\left\{0 \le i \le \min(m,n) : c_j = c'_j \text{ per ogni } 0 \le j \le i\right\}.$$
Se C denota l'intero $\frac{c - \sum_{i=0}^{e} c_i 2^i}{2^{e+1}}$, allora
$$C = \sum_{i=0}^{m-e-2} C_i 2^i = \sum_{j=0}^{n-e-2} C'_j 2^j,$$
con $C_i = c_{e+1-i}$ e $C'_i = c'_{e+1-i}$ sono due rappresentazioni NAF di C con $C_0 = c_{e+1} \neq c'_{e+1} = C'_0$. Quindi, possiamo assumere senza perdere di generalità che $c_0 \neq c'_0$. Eventualmente sostituendo c con $-c$, possiamo assumere che $c_0 = 1$. Allora c è dispari e quindi $c'_0 = -1$. Allora, $c_1 = c'_1 = 0$ essendo rappresentazioni NAF. Ma ciò è un assurdo, poiché $\sum_{i=0}^{m-1} c_i 2^i \equiv 1 \ mod\ 4$ mentre $\sum_{i=0}^{n-1} c'_i 2^i \equiv -1 \ mod\ 4$. Pertanto, vale l'unicità della rappresentazione NAF.

□

Esempio 12.11. $c = 3895 = 2^{11} + 2^{10} + 2^9 + 2^8 + 2^5 + 2^4 + 2^2 + 2^1 + 2^0$.

$$\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 \end{array}$$

Quindi la rappresentazione NAF di 3895 è $(1,0,0,0,-1,0,1,0,0-1,0,0,-1)$.

□

Sia P un punto di ordine n di una curva ellittica $\mathcal{E}$ su $GF(q)$ e sia $(c_{\ell-1},...,c_0)$ la rappresentazione binaria con segno di un intero c tale che $0 \le c \le n-1$. È possibile calcolare cP con una serie di raddoppi, addizioni e sottrazioni attraverso il seguente algoritmo.

Algoritmo 12.12. **(Raddoppi-Addizioni-Sottrazioni** $(P,(c_{\ell-1},\ldots,c_0))$**)**

$Q \longleftarrow \infty$
for $i \longleftarrow \ell - 1$ **downto** 0
do $\begin{cases} Q \longleftarrow 2Q \\ \textbf{if } c_i = 1 \\ \textbf{then } Q \longleftarrow Q + P \\ \textbf{else if } c_i = -1 \\ \textbf{then } Q \longleftarrow Q - P \end{cases}$
return (Q)

Si prova che, in media, il numero degli zeri presenti nella rappresentazione binaria e nella rappresentazione NAF di un intero di ℓ bit è $\ell/2$ e $2\ell/3$, rispettivamente. Pertanto,

- Se si utilizza la rappresentazione binaria di c nell'**Algoritmo 12.12**, il numero medio delle operazioni da eseguire $\ell+\ell/2$: ℓ raddoppi e $\ell/2$ addizioni.

- Se si utilizza la rappresentazione NAF di c nell'**Algoritmo 12.12**, il numero medio delle operazioni da eseguire $\ell + \ell/3$: ℓ raddoppi e $\ell/3$ tra addizioni e sottrazioni.

Quindi se assumiamo che il raddoppio impiega lo stesso tempo della somma o della sottrazione, il rapporto tra i tempi medi impiegati dallo stesso algoritmo nei due casi sopra descritti è

$$\frac{\ell + \ell/2}{\ell + \ell/3} = \frac{9}{8}.$$

Quindi, utilizzare la rappresentazioni NAF di un intero piuttosto che la rappresentazione binaria velocizza l'**Algoritmo 12.12** di circa 11%.

12.4.2 Algoritmo di Lenstra

In questo paragrafo descriviamo l'algoritmo randomizzato e basato sulle curve ellittiche dovuto a Lenstra, per fattorizzare un intero composito dispari n.

Figura 12.1: Hendrik Willem Lenstra Jr. (1949)

Come vedremo più avanti, il metodo di Lenstra è l'analogo del metodo $p-1$ di **Pollard**, basato sulle curve ellittiche.

Algoritmo 12.13. **($p-1$ di Pollard (n, B))**

$a \longleftarrow 2$
for $j \longleftarrow 2$ **to** B
do $\begin{cases} a \longleftarrow a^j \ \ mod\ n \\ d \longleftarrow \gcd(a^j \ \ mod\ n - 1, n) \end{cases}$
if $1 < d < n$
then return (d)
else return ("insuccesso")

Il principale limite del metodo di Pollard è che non è efficiente se $p-1$ è prodotto di potenze di primi piccoli, dove p è un divisore primo di n.

Definizione 12.14. Sia $m \in \mathbb{Z}$, allora per ogni $x_1, x_2 \in \mathbb{Q}$ scriveremo $x_1 \equiv_m x_2$ se, e solo se, $x_1 - x_2$ ridotto ai minimi termini ha numeratore divisibile per m.

Osservazione 12.15. Per ogni $x_1 \in \mathbb{Q}$ con denominatore primo con m esiste un unico $x_2 \in \mathbb{Z}$ tale che $0 \leq x_2 \leq m-1$ e $x_1 \equiv_m x_2$. Infatti, sia $x_1 \in \mathbb{Q}$, allora $x_1 = \frac{a}{b}$, con $mcd(a,b) = 1$. Siccome $mcd(b,m) = 1$, allora esiste un unico $0 \leq x_2 \leq m-1$ tale che $x_2 b \equiv_m a$. Quindi $m \mid a - x_2 b$ e, pertanto, $x_1 = \frac{a}{b} \equiv_m x_2$.

Sia $\mathcal{E}/\mathbb{Q} : y^2 = x^3 + ax + b$ una curva ellittica a coefficienti in $\mathbb{Q}$ e sia $n \in \mathbb{Z}$ tale che $mcd(\Delta, n) = 1$. Allora, per ogni punto $P = (x,y) \in \mathcal{E}/\mathbb{Q}$ le cui coordinate hanno denominatori primi con n e per ogni $p \in \mathbb{P}$, $p \mid n$ risulta $mcd(\Delta, p) = 1$, quindi è possibile considerare la curva $\mathcal{E}/GF(p) : y^2 = x^3 + ax + b$ a coefficienti in $GF(p)$. Inoltre, le coordinate del punto P hanno denominatori primi con p, pertanto $P \ mod\ p = (x \ mod\ p, y \ mod\ p)$, con $0 \leq x \ mod\ p, y \ mod\ p \leq p-1$ individuano un punto di $\mathcal{E}/GF(p)$.

Proposizione 12.16. Sia $\mathcal{E}/\mathbb{Q}$ una curva ellittica definita dall'equazione $y^2 = x^3 + ax + b$, dove $a, b \in \mathbb{Z}$ e $mcd(4a^3 + 27b^2, n) = 1$. Siano P_1, P_2 due punti su $\mathcal{E}/\mathbb{Q}$ le cui coordinate hanno denominatori primi con n e tali che $P_1 \neq -P_2$. Allora $P_1 + P_2 \in \mathcal{E}/\mathbb{Q}$ ha coordinate con denominatori primi con n se e solo se non esiste p primo tale che $p \mid n$ con la proprietà che $P_1 mod\ p + P_2 mod\ p = \infty mod\ p$.

Dimostrazione. Supponiamo dapprima che $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_1 \neq -P_2$ e $P_1 + P_2 \in \mathcal{E}$ abbiano coordinate con denominatori primi con n. Sia p un qualunque divisore primo di n. Dobbiamo dimostrare che $P_1 mod\ p + P_2 mod\ p \neq \infty mod\ p$.

Se $x_1 \not\equiv x_2 mod\ p$, allora $P_1 mod\ p \neq P_2 mod\ p$ e quindi posso applicare

$$\begin{cases} x_3 &= \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 &= -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3)\,. \end{cases} \tag{12.2}$$

e concludere che in questo caso $P_1 mod\ p + P_2 mod\ p \neq \infty mod\ p$.

Se invece $x_1 \equiv x_2 mod\ p$, ricordando che per ipotesi $P_1 \neq -P_2$, si possono verificare due casi: $P_1 = P_2$ oppure $P_1 \neq P_2$. Nel primo caso, essendo $P_1 = P_2$, allora le coordinate di $P_1 + P_2 = 2P_1$ sono date dalle formule

$$\begin{cases} x_3 &= \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 &= -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3)\,. \end{cases} \tag{12.3}$$

e $2P_1 mod\ p$ è dato dalle stesse formule con ciascun termine sostituito dal suo residuo $mod\ p$. Per mostrare che $P_1 mod\ p \neq \infty mod\ p$, basta dimostrare che il denominatore $2y_1$ che compare nelle formule (12.3) non è divisibile per p.
Procediamo per assurdo. Se $2y_1$ fosse divisibile per p, poiché per ipotesi il denominatore della prima coordinata del punto $2P_1$ non può essere divisibile per p, segue che dovrà essere divisibile per p il numeratore $3x_1^2 + a$. Ciò significa che x_1 è una radice in $GF(p)$ sia di $x^3 + ax + b$, che della sua derivata formale

$3x_1^2 + a$, contraddicendo la nostra ipotesi iniziale e cioè che non esistesse alcuna radice multipla di $x^3 + ax + b$ in $GF(p)$.

Ora supponiamo che $P_1 \neq P_2$. Poiché $x_1 \equiv x_2 mod\ p$ e $x_1 \neq x_2$, possiamo scrivere $x_2 = x_1 + p^r x$ con $r \geq 1$ scelto in modo tale che nè il numeratore nè il denominatore di x sia divisibile per p. Poiché abbiamo supposto che $P_1 + P_2$ abbia denominatore non divisibile per p, possiamo usare la formula (12.2) per concludere che y_2 è della forma $y_1 + p^r y$. Infatti, siccome $x_2 - x_1 = p^r x$, allora $\frac{y_2 - y_1}{x_2 - x_1}$ deve essere tale che, ridotto ai minimi termini, il denominatore non sia divisibile per p; segue che $y_2 - y_1 = p^r y$ (y può essere anche divisibile per p). Inoltre, poiché $P_2 \in \mathcal{E}/\mathbb{Q}$, vale che

$$\begin{aligned} y_2^2 &= (x_1 + p^r x)^3 + a(x_1 + p^r x) + b \\ &\equiv x_1^3 + ax_1 + b + p^r x(3x_1^2 + a)\ mod\ p^{r+1} \\ &= y_1^2 + p^r x(3x_1^2 + a)\ mod\ p^{r+1}. \end{aligned} \tag{12.4}$$

Poiché $x_2 \equiv x_1 mod\ p$ e $y_2 \equiv y_1 mod\ p$, segue che $P_1 mod\ p = P_2 mod\ p$ e quindi $P_1 mod\ p + P_2 mod\ p = 2P_1 mod\ p$ ma $2P_1 mod\ p = \infty mod\ p$ se e solo se $y_1 \equiv y_2 \equiv 0 mod\ p$. Se vale quest'ultima congruenza, significa che $y_1 + y_2 \equiv 0 mod\ p$ e ricordando che $y_1 \equiv y_2 mod\ p^r$, si ha che $y_1^2 - y_2^2 \equiv 0 mod\ p^{r+1}$. Pertanto, da (12.4), segue che $3x_1^2 + a \equiv 0 mod\ p$. Tuttavia ciò è impossibile perché il polinomio $x^3 + ax + b$ non ha radici multiple in $GF(p)$ e quindi x_1 non può essere radice sia di tale polinomio che della sua derivata. Pertanto possiamo concludere che $P_1 mod\ p + P_2 mod\ p \neq \infty mod\ p$.

Viceversa, supponiamo ora che per ogni p divisore primo di n, risulti $P_1 mod\ p + P_2 mod\ p \neq \infty mod\ p$ e mostriamo che, allora, $P_1 + P_2$ ha coordinate che, ridotte ai minimi termini, hanno denominatori primi con n.
Fissiamo p primo tale che $p \mid n$. Se $x_1 \not\equiv x_2\ mod\ p$, da (12.2) segue che p non divide il denominatore delle coordinate della somma. Se invece $x_1 \equiv x_2\ mod\ p$, allora $y_2 \equiv \pm y_1\ mod\ p$, ma poiché $P_1 mod\ p + P_2 mod\ p \neq \infty mod\ p$, dobbiamo avere $y_2 \equiv y_1 \not\equiv 0 mod\ p$. Pertanto, possono presentarsi due casi: $P_2 = P_1$ oppure $P_2 \neq P_1$.

Nel primo caso $y_1 \not\equiv 0 mod\ p$ poiché $2P_1 \neq \infty\ mod\ p$ per via della formula (12.3). Quindi le coordinate di $P_1 + P_2 = 2P_1$ hanno denominatore primo con p. Infine, nel secondo caso, scriviamo nuovamente $x_2 = x_1 + p^r x$ con x non divisibile per p. Da (12.4) segue che

$$y_2^2 \equiv y_1^2 + p^r x(3x_1^2 + a) mod\ p$$

e ricordando che $p^r x = x_2 - x_1$, otteniamo

$$\frac{y_2^2 - y_1^2}{x_2 - x_1} \equiv 3x_1^2 + a\ mod\ p.$$

Si noti che in questa congruenza il membro a destra ha denominatore primo con p. Ora, poiché p non divide $y_1 + y_2$, essendo $y_1 + y_2 \equiv 2y_1 mod\ p$, possiamo dividere per $y_1 + y_2$, ottenendo cosí

$$\frac{y_2^2 - y_1^2}{(x_2 - x_1)(y_2 + y_1)} = \frac{y_2 - y_1}{x_2 - x_1}$$

ed anche questo termine ha denominatore primo con p. Ricordando che le formule per calcolare le coordinate di P_1+P_2 sono date da (12.2), possiamo concludere che tali coordinate hanno denominatore non divisibile per p. Dall'arbitrarietà con cui è stato scelto p divisore primo di n, segue che $P_1 + P_2$ ha coordinate prime con n.

□

Esempio 12.17. Poiché vale $\Delta = 4+27\times6^2 = 976 \neq 0$, allora ha senso considerare la curva ellittica sui razionali

$$\mathcal{E}/\mathbb{Q} : y^2 = x^3 + x + 6$$

Applichiamo la **Proposizione 12.16** per fattorizzare $n = 1763$.
Vale che

$$mcd(\Delta, n) = mcd(976, 1763) = 1.$$

Basta sostituire nell'equazione della curva per verificare che $P_1 = (2,4)$ e $P_2 = \left(-\frac{87}{64}, \frac{747}{512}\right)$ appartengono alla curva ellittica $\mathcal{E}/\mathbb{Q}$ e hanno coordinate i cui denominatori sono primi con n. Inoltre, vale che $P_2 \neq -P_1$. Siccome $P_1 \neq P_2$, utilizzando la formula (12.2), si ha che $P_1 + P_2 = (x_3, y_3)$ con

$$\begin{cases} x_3 = \left(\frac{\frac{747}{512}-4}{-\frac{87}{64}-2}\right)^2 - 2 + \frac{87}{64} = -\frac{3166}{46\,225} = -\frac{2\times1583}{5^2\times43^2} \\ y_3 = -4 + \left(\frac{\frac{747}{512}-4}{-\frac{87}{64}-2}\right)\left(2 - \left(-\frac{3166}{46\,225}\right)\right) = -\frac{24\,203\,948}{9938\,375} = -\frac{2^2\times19\times318\,473}{5^3\times43^3} \end{cases}$$

Consideriamo $p = 43$. Allora $P_1 mod 43 = (2,4)$ e $P_2 mod 43 = (2,39)$ e quindi, avendo la stessa ascissa, segue che

$$P_1 \ mod\ 43 + P_2 \ mod\ 43 = \infty \ mod\ 43.$$

Quindi, applicando la **Proposizione 12.16**, vale che 43 divide $n = 1763$. Questo è vero, infatti

$$n = 1763 = 41 \times 43.$$

□

L'algoritmo opera come di seguito esposto.

1. Si generi la coppia $(\mathcal{E}, P)$, dove $\mathcal{E}$ è una curva ellittica definita dall'equazione $y^2 = x^3 + ax + b$ con $a, b \in \mathbb{Z}$ e $P = (x, y)$ è un suo punto. Si fissi inoltre un intero positivo h come numero massimo di curve da generare.
2. Si calcoli $m = (4a^3 + 27b^2, n)$.
   - Se $m = 1$, si vada al passo successivo;
   - Se $1 < m < n$, è stato trovato un divisore non banale di n, pertanto l'algoritmo termina;
   - Se $m = n$, si ritorni al punto (1) e si scelga un'altra coppia $(\mathcal{E}, P)$.
3. Si costruisca
$$k = \prod_{\substack{l \leq B \\ l \text{ primo}}} l^{\alpha_l} \tag{12.5}$$
con $\alpha_l = \left[\frac{logC}{logl}\right]$ dove B e C sono due interi positivi che rappresentano rispettivamente un limite per i primi l ed un limite per le potenze di tali primi che compaiono nella fattorizzazione di k. B e C devono essere sufficientemente grandi in modo tale da aumentare la probabilità di trovare un divisore di n e limitare i tempi previsti per il calcolo.
4. Si calcoli kP con il metodo dei raddoppi successivi (come abbiamo visto il calcolo si può accelerare di circa l' 11% utilizzando la rappresentazione NAF di k).

Ad ogni iterazione si utilizzino le formule (11.21) o (11.22) cioè le formule

$$\begin{cases} x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \\ y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3)\,. \end{cases} \tag{12.6}$$

oppure

$$\begin{cases} x_3 = \left(\frac{3x_1^2 + A}{2y_1}\right)^2 - 2x_1 \\ y_3 = -y_1 + \left(\frac{3x_1^2 + A}{2y_1}\right)(x_1 - x_3)\,. \end{cases} \tag{12.7}$$

a seconda che i punti da sommare siano diversi o uguali, rispettivamente. In particolare, in ogni passaggio, viene utilizzato l'algoritmo euclideo per trovare gli inversi $mod\ n$ di $x_2 - x_1$ o $2y_1$, rispettivamente.

Se in un qualsiasi passo l'algoritmo euclideo fallisce nell'intento di trovare l'inverso $mod\ n$ richiesto, allora detto m il massimo comun divisore tra n ed il denominatore di (12.6) o di (12.7) rispettivamente, si presentano due possibilità:

(i) Se m è tale che $1 < m < n$, allora è stato trovato un divisore non banale di n e quindi l'algoritmo termina;

(ii) Se $m = n$ allora si ritorna al passo **(1)** e si ripete il procedimento descritto dall'algoritmo per una nuova coppia $(\mathcal{E}, P)$.

Per la **Proposizione 12.16**, le possibilità **(i)** e **(ii)** si verificano quando esiste un multiplo k_iP tale che $k_iP mod\ p = \infty mod\ p$ per qualche divisore primo p di n o per tutti i divisori primi di n.

Quindi il metodo di Lenstra determina un fattore non banale di n, oppure, quando esso fallisce, un punto di $\mathcal{E}$. Se non riusciamo a trovare un divisore non banale di n, l'algoritmo termina dopo h fallimenti, dove h rappresenta il numero massimo di curve che possiamo considerare.

Se la probabilità di fallire è $0 < \rho < 1$, allora l'**Algoritmo di Lenstra** è un $(h, 1-\rho^h)$ algoritmo di tipo Las Vegas e quindi

$$\lim_{h\to\infty} 1 - \rho^h = 1.$$

Pertanto, con un'alta probabilità, riusciremo a fattorizzare n in un numero ragionevole di tentativi.

Esempio 12.18. Utilizziamo il metodo di Lenstra per fattorizzare $n = 5429$.

Consideriamo la famiglia di curve ellittiche descritta dall'equazione

$$y^2 = x^3 + ax - a$$

con $a = 1, 2, \ldots$. Osserviamo che ogni curva ellittica che appartiene a questa famiglia, contiene il punto $P = (1, 1)$. Dopo aver verificato che $(4a^3 + 27a^2, 5429) = 1$, scegliamo $B = 3$ e $C = 92$ e quindi, tenendo conto di questi limiti, consideriamo $k = 2^6 3^4$. Per ciascun valore di a, calcoliamo kP con il metodo dei raddoppi successivi.

1. Per $a = 1$, risulta che $3^4 2^6 P mod\ p$ è un punto di $\mathcal{E} mod\ p$ per ogni $p \mid n$.
2. Per $a = 2$ troviamo che, quando cerchiamo di calcolare $3^2 2^6 P mod\ p$, otteniamo un denominatore f tale che $(f, 5429) = 61$. Questo massimo comun divisore rappresenta uno dei fattori propri di 5429. Cioè il punto $(1, 1)$ ha ordine che divide $3^2 2^6$ sulla curva $y^2 = x^3 + 2x - 2 mod\ 61$. Pertanto, il nostro secondo tentativo ha avuto successo.
3. Per $a = 3$, quando cerchiamo di calcolare $3^4 2^6 P$, otteniamo l'altro fattore primo di 5429 che è 89. Pertanto abbiamo ottenuto che $5429 = 61 \cdot 89$.

□

Osservazione 12.19. È evidente la somiglianza di questo metodo con il metodo di Pollard, sebbene, mentre in quest'ultimo si usa il gruppo $\mathbb{Z}_p^*$, qui invece utilizziamo il gruppo costituito dai punti di $\mathcal{E} mod\ p$. Il fatto che il metodo di Pollard dipende dal gruppo $\mathbb{Z}_p^*$ (più precisamente, dai vari gruppi di questo tipo, poiché p varia tra i divisori primi di n), costituisce un limite di tale metodo. Infatti, per un fissato n, i gruppi del tipo $\mathbb{Z}_p^*$, sono fissati. Se accade che tutti i gruppi $\mathbb{Z}_p^*$ hanno ordine divisibile per un primo elevato, ciò può costituire un problema.

Quindi la differenza principale tra i due metodi sta nel fatto che, **mentre nel metodo di Pollard il gruppo utilizzabile è soltanto uno, nel metodo di Lenstra, lavorando con le curve ellittiche su $GF(p)$, abbiamo una serie di gruppi da usare e realisticamente possiamo sperare sempre di trovare un gruppo il cui ordine non sia divisibile per un primo grande o per una sua potenza.** Infatti, se la curva $\mathcal{E}$ non va bene, cioè per ogni $p \mid n$ il gruppo $\mathcal{E} mod\ p$ ha ordine divisibile per un primo grande (e quindi è improbabile che $kP mod\ p$ sia uguale a $\infty mod\ p$ per k dato da (12.5), possiamo effettuare un'altra scelta della curva $\mathcal{E}$ e del punto $P \in \mathcal{E}$.

Il metodo di Lenstra presenta alcuni vantaggi rispetto ad altri algoritmi di fattorizzazione:

1. Esso è il solo metodo che è sostanzialmente più veloce se n è divisibile per un primo che è molto più piccolo di $\sqrt{n}$. Infatti, dal **Teorema di Hasse**, segue che quanto più è piccolo il divisore p di n, minore sarà il numero di punti sulla curva ellittica che stiamo considerando. Ciò comporta un minor numero di operazioni da effettuare per calcolare kP. Tale aspetto risulta rilevante quando la scelta della curva non è ottimale. In tal caso, infatti, un valore di p piccolo, consente di effettuare più velocemente i calcoli sulla curva data e quindi di passare in breve tempo ad una nuova scelta della coppia $(\mathcal{E}, P)$.

2. Dal **Teorema di Hasse** segue che il gruppo formato dai punti di una curva ellittica ha ordine del tipo $p + 1 - t_p$ dove $t_p \leq 2\sqrt{p}$; se per qualche divisore primo p di n il numero $p+1-t_p$ è costituito da fattori primi piccoli, allora l'algoritmo fornisce un divisore non banale di n, altrimenti no.

3. Come già visto nell'**Osservazione 12.19**, il metodo di Lenstra, qualora dovesse fallire con una determinata scelta di $(\mathcal{E}, P)$, permette di ritentare la fattorizzazione utilizzando altre coppie del tipo $(\mathcal{E}_i, P_i)$.

4. Il metodo descritto può essere utilizzato in combinazione con altri metodi quando è richiesta la fattorizzazione di alcuni numeri ausiliari.

13 Firma Digitale

13.1 Sistemi di Firma

Un sistema di firma è un metodo per firmare i messaggi in formato elettronico in modo da poterli trasmettere attraverso una rete di computer. Le differenze sostanziali con la firma convenzionale sono:

1. La firma convenzionale è allegata fisicamente al documento fisico da firmare, invece quella digitale è collegata in modo opportuno al documento elettronico attraverso un opportuno algoritmo.

2. La verifica della firma convenzionale avviene attraverso la comparazione con quella originale (ciò la rende facilmente falsificabile), invece quella digitale usando un algoritmo pubblico noto (ciò la rende, con le dovute attenzioni, difficile da falsificare).

3. Nel caso della firma convenzionale una copia del un documento firmato è distinguibile dall'originale. Contrariamente alla firma convenzionale, una copia della firma digitale è perfettamente identica all'originale. Quindi, nel caso della firma digitale bisogna prevenire che questa venga riutilizzata maliziosamente.

Forniamo una definizione formale di un sistema di firma

Definizione 13.1. **(Sistema di Firma)**
Un **Sistema di Firma** è un quintupla $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ di insiemi finiti e non vuoti soddisfacenti alle seguenti condizioni:

1. $\mathcal{P}$ è l'insieme dei possibili **messaggi**.
2. $\mathcal{A}$ è l'insieme delle possibili **firme**.
3. $\mathcal{K}$ è l'insieme delle possibili **chiavi**.
4. Per ogni $K \in \mathcal{K}$ ci sono un **algoritmo di firma** $\text{sig}_K \in \mathcal{S}$ ed un corrispondente **algoritmo di verifica** $\text{ver}_K \in \mathcal{V}$, dove

$$\begin{aligned} \text{sig}_K &: \mathcal{P} \longrightarrow \mathcal{A} \\ \text{ver}_K &: \mathcal{P} \times \mathcal{A} \longrightarrow \{vero, falso\} \end{aligned}$$

sono funzioni tali che per $(x, y) \in \mathcal{P} \times \mathcal{A}$ vale che

$$\text{ver}_K(x, y) = vero \iff y = \text{sig}_K(x).$$

La coppia (x, y) è detta **messaggio firmato**.

Fissato $K \in \mathcal{K}$ devono valere le seguenti proprietà:

- sig_K e ver_K devono avere complessità computazionale polinomiale, quindi devono essere efficienti come algoritmi.
- L'algoritmo sig_K è **segreto**, invece ver_K è **pubblico**. Inoltre deve essere computazionalmente difficile per un potenziale avversario di determinare una firma y tale che $\text{ver}_K(x, y) = vero$ (si noti che ci possono essere più di una firma valida y per un fissato messaggio x).
- Se un potenziale avversario è in grado di determinare (x, y) con $\text{ver}_K(x, y) = vero$ e x non precedentemente firmato da una delle parti oneste, allora y è detta **falsificazione**.

Il seguente esempio mostra come il crittosistema RSA possa essere utilizzato per fornire un sistema di firma digitale.

Definizione 13.2. **(Sistema di Firma RSA)**
Siano $n = pq$ con p, q primi distinti. Siano $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$ e

$$\mathcal{K} = \{(n, p, q, d, e) : n = pq \text{ e } ed \equiv 1 \ mod \ \varphi(n)\}$$

allora (n, e) sono pubblici e (p, q, d) privati.
Fissato $K = (n, p, q, e, d)$, per ogni $x, y \in \mathbb{Z}_n$ definiamo

$$\begin{aligned} \text{sig}_K(x) &= x^d \ mod \ n \\ \text{ver}_K(x, y) &= vero \iff x \equiv y^e \ mod \ n. \end{aligned}$$

1. L'utente A firma un messaggio il messaggio x con la funzione di decifratura d_{K_A}, essendo $\text{sig}_{K_A} = d_{K_A}$. È l'unica persona che può creare la firma essendo $\text{sig}_{K_A} = d_{K_A}$ privata.
2. A trasmette (x, y), dove $y = d_{K_A}(x)$ all'utente B
3. Una volta ricevuto il messaggio firmato (x, y), l'utente B utilizza la chiave pubblica $e_{K_A} = \text{ver}_{K_A}$ per calcolare $x = e_{K_A}(y)$.

La firma dell'utente A può essere 'falsificata' come segue dall'avversario C. Fissa y e calcola $x = e_{K_A}(y)$ essendo e_{K_A} pubblica. Quindi, y è una firma valida per il messaggio x, infatti $x = d_{K_A}(y)$. Tuttavia la probabilità che il messaggio x abbia un significato è molto bassa. Viceversa, per determinare un messaggio firmato (x, y) partendo da x bisogna violare il crittosistema RSA.

13.2 Combinare la firma digitale e la crittografia a chiave pubblica

L'utente A vuole trasmettere un messaggio all'utente B.

- A **firma e poi cifra** il messaggio da trasmettere a B (metodo raccomandato).

  1. L'utente A firma un messaggio x con la funzione di decifratura d_{K_A}, essendo $\text{sig}_{K_A} = d_{K_A}$. Quindi il messaggio firmato è $(x, d_{K_A}(x))$.
  2. A cifra $(x, d_{K_A}(x))$ con la funzione di cifratura pubblica e_{K_B} di B, e trasmette a B la coppia $(e_{K_B}(x), e_{K_B}(d_{K_A}(x)))$.
  3. Una volta ricevuto il messaggio prima firmato e poi criptato firmato $(e_{K_B}(x), e_{K_B}(d_{K_A}(x)))$, l'utente B utilizza la sua firma segreta d_{K_B} per recuperare $(x, d_{K_A}(x))$ e poi la funzione di cifratura pubblica di e_{K_A} per verificare la firma di A, ovvero che $x = e_{K_A}(d_{K_A}(x))$.

- A **cifra e poi firma** il messaggio da trasmettere a B (metodo non raccomandato).

  1. L'utente A cifra un messaggio x con e_{K_B} attraverso la chiave pubblica di B e successivamente lo firma con la sua chiave segreta d_{K_A}. Quindi, trasmette a B il messaggio cifrato e firmato $(e_{K_B}(x), d_{K_A}(e_{K_B}(x)))$.
  2. Una volta ricevuto il messaggio, l'utente B usa prima la propria chiave privata d_{K_B} ottenendo così $(x, d_{K_A}(x))$ $(e_{K_B}(x), d_{K_A}(e_{K_B}(x)))$.
  3. Infine, B usa la funzione di cifratura pubblica di e_{K_A} per verificare $x = e_{K_A}(d_{K_A}(x))$.

Il secondo metodo non è raccomandato perché è suscettibile del seguente attacco:

1. L'utente A cifra un messaggio x con e_{K_B} attraverso la chiave pubblica di B e successivamente lo firma con la sua chiave segreta d_{K_A}. Quindi, trasmette a B il messaggio cifrato e firmato $(e_{K_B}(x), d_{K_A}(e_{K_B}(x)))$.

2. Un utente malizioso C, intercettato $(e_{K_B}(x), d_{K_A}(e_{K_B}(x)))$, trasmette $(e_{K_B}(x), d_{K_C}(e_{K_B}(x)))$ all'utente B.

3. Una volta ricevuto il messaggio, l'utente B usa prima la propria chiave privata d_{K_B} ottenendo così $(x, d_{K_C}(x))$. C intercetta il messaggio $(x, d_{K_C}(x))$ e successivamente calcola $x = e_{K_C}(d_{K_C}(x))$. Quindi, B ne deduce che il testo in chiaro x, che C non conosce, è stato trasmesso da C.

13.3 Requisiti di sicurezza per i sistemi di firma

In questa sezione analizziamo quando un sistema si firma è 'sicuro'. L'analisi vine fatta rispetto ai seguenti modelli di attacco:

- **Attacco basato sulla conoscenza della chiave.**
  L'avversario solo conosce la chiave pubblica ver_K.
- **Attacco basato sulla conoscenza di messaggi.**
  L'avversario conosce una lista di messaggi firmati $(x_1, y_1), ..., (x_q, y_q)$ da un utente, i.e $y_i = \text{sig}_K(x_i)$ con $i = 1, ..., q$.
- **Attacco basato sulla conoscenza di messaggi scelti.**
  L'avversario chiede conosce una lista di messaggi firmati $(x_1, y_1), ..., (x_q, y_q)$ da un utente, i.e $y_i = \text{sig}_K(x_i)$ con $i = 1, ..., q$, in cui $x_1, ..., x_q$ sono messaggi scelti dallo stesso stesso avversario.

I possibili obiettivi dell'avversario sono:

- **Decifratura totale.**
  L'avversario determina la funzione sig_K, quindi crea firme valide su un qualsiasi messaggio.
- **Falsificazione selettiva.**
  La probabilità con cui un avversario è capace di creare una firma valida su un messaggio scelto da un (altro) utente non è trascurabile. Cioè, se x è un messaggio scelto da un utente, allora un avversario produce una firma y per x tale che
  $$P[(x, y) : \text{ver}_K(x, y) = vero] >> 0.$$
- **Falsificazione esistenziale.**
  L'avversario crea una firma valida per almeno un messaggio. Cioè crea un a coppia (x, y) dove x messaggio non precedentemente firmato da un altro utente e $\text{ver}_K(x, y) = vero$.

I sistemi di firma non sono incondizionatamente sicuri poiché per un qualsiasi messaggio x un avversario testa tutti i possibili $y \in \mathcal{A}$ fino a che non trova che $\text{ver}_K(x, y) = vero$, essendo ver_K pubblico. Quindi, avendo un tempo sufficiente a disposizione, un avversario può sempre falsificare una firma. Pertanto, l'obiettivo è costruire sistemi di firma che sono computazionalmente sicuri o dimostrabilmente sicuri.

Vediamo i concetti sopra descritti applicati al Sistema di Firma RSA

- **Falsificazione esistenziale basata sulla sola conoscenza della chiave (pubblica):**
  L'avversario sceglie una firma y e calcola $x = e_K(y)$ essendo e_K pubblica. Quindi, y è una firma valida per il messaggio x, infatti $x = d_K(y)$, i.e. $\text{ver}_K(x, y) = vero$.

- **Falsificazione esistenziale basata sulla conoscenza di messaggi:**
  L'avversario i messaggi firmati $(x_1, y_1), (x_2, y_2)$, quindi $y_i = x_i^d \ mod \ n$, $i = 1, 2$, allora $y_1 y_2 = x_1^d x_2^d \ mod \ n$ è una firma valida per il messaggio $x_1 x_2$.

- **Falsificazione selettiva basata sulla conoscenza di messaggi scelti:**
  L'avversario vuole creare la firma per un messaggio x scelto da un utente. Allora determina $x_1, x_2 \in \mathbb{Z}_n$ tali che $x = x_1 x_2 \ mod \ n$ e chiede all'utente di firmare x_1 e x_2. Quindi, procedendo come nel caso precedente, se $y_i = x_i^d \ mod \ n$, $i = 1, 2$, allora $y_1 y_2 = x^d \ mod \ n$ è una firma valida per il messaggio x.

13.4 Sistemi di firma e funzioni Hash

I sistemi di firma sono quasi sempre usati congiuntamente ad una funzione hash crittografica pubblica molto veloce.

La funzione hash $h : \{0,1\}^* \longrightarrow \mathcal{Z}$ dove $\mathcal{Z}$ è un insieme di stringhe di lunghezza binaria fissata, generalmente di 160 bit e il sistema di firma $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ è tale che $\mathcal{Z} \subseteq \mathcal{P}$. Quindi un utente invece di firmare un messaggio, firma il suo sunto. Gli utenti applicano il seguente protocollo:

1. L'utente A sceglie un messaggio x e ne calcola il sunto $z = h(x)$.
2. A sceglie una chiave segreta K e firma z, cioè calcola $y = \text{sig}_K(z)$, e successivamente trasmette la coppia (x, y).
3. Una volta ricevuto (x, y), il destinatario prestabilito B calcola $z = h(x)$, essendo h pubblica e successivamente verifica la firma di A, cioè calcola $\text{ver}_K(z, y) = vero$.

E' importante prestare attenzione all'utilizzo della funzione hash h al fine di non indebolire la sicurezza del sistema di firma $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$. Generalmente, perchè la sicurezza del sistema di firma sia preservata è sufficiente che h sia resistente rispetto ai problemi dell'immagine inversa, della seconda immagine inversa, o della collisione al fine. Infatti,

- **Le falsificazioni esistenziali usando un attacco basato sulla conoscenza di messaggi sono impedite se h è resistente alla seconda immagine inversa.**
  Se così non fosse e un avversario ha sia un messaggio firmato (x, y), i.e. $y = \text{sig}_K(h(x))$ da un generico utente e sia inoltre capace di terminare $x' \neq x$ tale che $h(x') = h(x)$, allora esso determina la falsificazione (x', y) essendo $y = \text{sig}_K(h(x)) = \text{sig}_K(h(x'))$.

- **Le falsificazioni esistenziali usando un attacco basato sulla conoscenza di messaggi scelti sono impedite se h è resistente alla collisione.**
  In questo caso l'avversario determina x, x' distinti e tali che $h(x) = h(x')$. Quindi, chiede ad un utente di firmare x, ottenendo $y = \text{sig}_K(h(x))$. Allora genera la falsificazione (x', y).

- **Le falsificazioni esistenziali usando un attacco basato sulla conoscenza della chiave (pubblica) sono impedite se h è resistente all'immagine inversa.**
  Se l'avversario è capace di calcolare la firma y ad un elemento di $\mathcal{Z}$ ed è in grado di determinare un messaggio x tale che $z = h(x)$, allora (x, y) è una falsificazione.

13.5 Sistema di Firma di ElGamal

Il seguente sistema di Firma dovuto ad ElGamal non è deterministico. Quindi ci sono più firme valide per un qualsiasi messaggio e l'algoritmo di verifica delle firme le deve accettare tutte.

Definizione 13.3. **(Sistema di Firma di ElGamal (1985))**
Sia p un primo tale che il PLD sia intrattabile in $\mathbb{Z}_p^*$ e sia α un elemento primitivo di $\mathbb{Z}_p^*$. Siano $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ e

$$\mathcal{K} = \left\{(p, \alpha, d, \beta) : \beta \equiv \alpha^d \ mod\ p\right\}$$

Allora, la terna (p, α, β) è pubblica, invece d è segreto.
Per ogni $K = (p, \alpha, d, \beta)$ e per ogni random $k \in \mathcal{U}(\mathbb{Z}_{p-1})$, si definisce

$$\text{sig}_K(x, k) = (\gamma, \delta)$$

dove

$$(\gamma, \delta) = (\alpha^k \ mod\ p, (x - d \times \alpha^k \ mod\ p)k^{-1} \ mod\ (p-1)).$$

Per ogni $x, \gamma \in \mathbb{Z}_p^*$ e $\delta \in \mathbb{Z}_{p-1}$ definiamo

$$\text{ver}_K(x, (\gamma, \delta)) = vero \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \ mod\ p.$$

Proposizione 13.4. Vale che

$$\mathrm{ver}_K(x,(\gamma,\delta)) = vero \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \ mod\ p.$$

Dimostrazione. Supponiamo che $\mathrm{ver}_K(x,(\gamma,\delta)) = vero$, dove

$$(\gamma,\delta) = (\alpha^k \ mod\ p, (x - d \times \alpha^k \ mod\ p)k^{-1} \ mod\ (p-1)).$$

Allora

$$\begin{aligned} \beta^\gamma \gamma^\delta &= \alpha^{d(\alpha^k \ mod\ p)} \alpha^{k[\theta(p-1)+(x-d\times\alpha^k \ mod\ p)k^{-1}]} \\ &= \alpha^{d(\alpha^k \ mod\ p)} \alpha^{k\theta(p-1)} \alpha^x \alpha^{-d(\alpha^k \ mod\ p)} \\ &= \alpha^{k\theta(p-1)} \alpha^x \\ &= \alpha^x. \end{aligned}$$

Se $\beta^\gamma \gamma^\delta \equiv \alpha^x \ mod\ p$ e γ è un generatore della parte moltiplicativa del campo, allora esistono interi d e k tali $\beta \equiv \alpha^d \ mod\ p$ e $\gamma \equiv \alpha^k \ mod\ p$.

$$\alpha^{d(\alpha^k \ mod\ p)} \alpha^{k\delta} \equiv \alpha^x \ mod\ p$$

e quindi

$$\alpha^{d(\alpha^k \ mod\ p)+k\delta} \equiv \alpha^x \ mod\ p$$

che è equivalente a

$$x = (d\,(\alpha^x \ mod\ p) + k\delta) \ mod\ (p-1)$$

ovvero, tenendo presente che $k \in \mathcal{U}(\mathbb{Z}_{p-1})$,

$$\delta \equiv (x - d \times \alpha^k \ mod\ p)k^{-1} \ mod\ (p-1).$$

Quindi, $\mathrm{sig}_K(x) = (\gamma,\delta)$, cioè $\mathrm{ver}_K(x,(\gamma,\delta)) = vero$.

□

Esempio 13.5. Sia $p = 467$, $\alpha = 2$, $d = 127$ e quindi

$$\beta = 2^{127} \ mod\ 467 = 132$$

Supponiamo che un utente A voglia firmare il messaggio $x = 100$. Quindi, sceglie un intero casuale $k = 213$. Siccome $\gcd(213, 466) = 1$, allora $k \in \mathcal{U}(\mathbb{Z}_{466})$ e quindi

$$k^{-1} \ mod\ 466 = 431.$$

Ne segue che

$$\begin{aligned} \gamma &= 2^{431} \ mod\ 467 = 29 \\ \delta &= ((100 - 127 \times 29) \times 431) \ mod\ 466 = 51 \end{aligned}$$

Quindi

$$\mathrm{sig}_{(467,2,127,132)}(100) = (29,51)$$

siccome $132^{29}29^{51} \equiv 2^{100} \ mod\ 467$, allora

$$\mathrm{ver}_{(467,2,127,132)}(100,(29,51)) = vero.$$

□

13.6 Sicurezza del Sistema di Firma di ElGamal

Supponiamo che un avversario voglia falsificare la firma del generico messaggio senza conoscere d.

1. L'avversario sceglie x, γ e vuole determinare $\delta = \log_\gamma \alpha^x \beta^{-\gamma}$ deve risolvere il PLD in $\mathbb{Z}_p^*$.

2. L'avversario sceglie x, δ e vuole determinare γ allora deve risolvere l'equazione $\beta^\gamma \gamma^\delta \equiv \alpha^x \ mod\ p$ che è risulta difficile da trattare pur non essendo stata ampiamente studiata come al PLD.

3. L'avversario sceglie γ, δ e vuole determinare $x = \log_\alpha \beta^\gamma \gamma^\delta$ in $\mathbb{Z}_p^*$.

4. Falsificazione esistenziale basata sulla sola conoscenza della chiave pubblica: l'avversario vuole determinare γ, δ e x. Ciò è possibile, se $\gamma = \alpha^i \beta^j$ in $\mathbb{Z}_p^*$ con $0 \le i, j \le p-2$ con $\gcd(j, p-1) = 1$, allora

$$\begin{cases} \gamma = \alpha^i \beta^j \ mod\ p \\ \delta = -\gamma j^{-1} \ mod\ (p-1) \\ x = -\gamma i j^{-1} \ mod\ (p-1) \end{cases}$$

   implica

$$\begin{cases} x - i\delta \equiv 0 \ mod\ (p-1) \\ \gamma + j\delta \equiv 0 \ mod\ (p-1) \end{cases}$$

   Pertanto

$$\begin{aligned} \alpha^{x-i\delta} \equiv \beta^{\gamma+j\delta} \ mod\ p &\iff \alpha^x \equiv \beta^\gamma (\alpha^i \beta^j)^\delta \ mod\ p \\ &\iff \alpha^x \equiv \beta^\gamma \gamma^\delta \ mod\ p \end{aligned}$$

   e quindi

$$\mathrm{ver}_K(x, (\gamma, \delta)) = vero.$$

   Così, $\mathrm{sig}_K(x) = (\gamma, \delta)$ ovvero (γ, δ) è una firma valida per il messaggio x.

5. Falsificazione esistenziale basata sulla sola conoscenza di un messaggio firmato: l'avversario vuole determinare una firma valida (γ', δ') per un messaggio x' a partire dal messaggio x con firma (γ, δ).

   Siano h, i, j interi tali che $0 \le h, i, j \le p-2$ e $\gcd(h\gamma - j\delta, p-1) = 1$ e siano

$$\begin{cases} \gamma' = \gamma^h \alpha^i \beta^j \ mod\ p \\ \delta' = \delta\lambda(h\gamma - j\delta)^{-1} \ mod\ (p-1) \\ x' = \gamma^h \alpha^i \beta^j (hx + i\delta)(h\gamma - j\delta)^{-1} \ mod\ (p-1) \end{cases}$$

   È facile vedere che e che quindi (γ', δ') è firma valida per un messaggio x'.

6. La conoscenza da parte dell'avversario dell'intero casuale intero casuale k invertibile nelle classi di resto mod $p-1$ e di un messaggio firmato $(x,(\gamma,\delta))$ implica la conoscenza della chiave segreta

$$d = (x - k\delta)\gamma^{-1}$$

a questo punto l'avversario può creare firme valide a volontà.

7. L'utilizzo dello stesso intero casuale k per determinare per firmare due messaggi distinti implica la determinazione di k e quindi della chiave segreta d.

Supponiamo che (γ,δ_i) sia la firma di x_i, $i = 1,2$. Quindi valgono $\beta^\gamma\gamma^{\delta_1} \equiv \alpha^{x_1} \ mod\ p$ e $\beta^\gamma\gamma^{\delta_2} \equiv \alpha^{x_2} \ mod\ p$.

$$\gamma^{\delta_1-\delta_2} \equiv \alpha^{x_1-x_2} \ mod\ p \Longleftrightarrow \alpha^{k(\delta_1-\delta_2)} \equiv \alpha^{x_1-x_2} \ mod\ p$$

che è equivalente a

$$k(\delta_1 - \delta_2) \equiv (x_1 - x_2) \ mod\ (p-1) \tag{13.1}$$

Sia $D = \gcd(\delta_1 - \delta_2, p-1)$.

Siccome $D \mid \delta_1 - \delta_2$ e $D \mid p-1$, allora $D \mid x_1 - x_2$. Siano

$$\begin{cases} x' = \frac{x_1-x_2}{D} \\ \delta' = \frac{\delta_1-\delta_2}{D} \\ p' = \frac{p-1}{D} \end{cases}$$

allora (13.1) diventa

$$x' \equiv k\delta' \ mod\ p'.$$

Siccome $\gcd(\delta', p') = 1$, allora sia $\varepsilon = \delta'^{-1} \ mod\ p'$ e quindi

$$k = x'\varepsilon \ mod\ p'.$$

Pertanto,

$$k = (x'\varepsilon + ip') \ mod\ (p-1)$$

con $0 \le i \le p'-1$ essendo $p' = \frac{p-1}{D}$. Dei p' candidati valori di k solo uno è corretto che realizza $\gamma \equiv \alpha^k \ mod\ p$ e per tali valore come visto sopra $d = (x_1 - k\delta_1)\gamma^{-1}$.

- Gli attacchi **(1)** – **(3)** sono impediti scegliendo un primo p tale che il PLD sia intrattabile in $\mathbb{Z}_p^*$.
- Per quanto visto, gli attacchi **(4)** e **(5)** sono impediti associando al Sistema di Firma di ElGamal un funzione hash che sia resistente all'immagine inversa e alla seconda immagine inversa, rispettivamente.
- Gli attacchi **(6)** e **(7)** sono impediti mantenendo segreto l'intero random k e usandone uno diverso per ogni messaggio da firmare.

13.7 Varianti del Sistema di firma ElGamal

In molte situazioni un messaggio è cifrato e decifrato solo una volta, quindi è importante che il crittosistema sia sicuro solo al momento della cifratura del messaggio. Un messaggio firmato, invece, ha un valore legale come un contratto o un testamento quindi è necessario verificare la firma anche molti anni dopo che il messaggio è stato firmato. Quindi, bisogna avere molte più precauzioni su un sistema di firma piuttosto che ad un crittosistema. Siccome il sistema di firma di ElGamal fonda la sua sicurezza sul PLD, affinchè questa sia duratura nel nel tempo il modulo p deve essere costituito da 1024 bit (in realtà anche più grande) questo significa che la firma nel sistema di ElGamal è di almeno 2048 bit.

A fine di di rendere la firma notevolmente più corta sono state progettati i seguenti sistemi analizzati singolarmente in seguito:

1. Sistema di Firma di Schnorr.
2. L'algoritmo di firma digitale (DSA).
3. L'algoritmo di firma digitale basato sulle curve ellittiche (ECDSA).

13.7.1 Il Sistema di Firma di Schnorr

Figura 13.1: Claus-Peter Schnorr (1943)

Definizione 13.6. **(Sistema di firma di Schnorr (1990))**
Siano p un primo tale che il PLD sia intrattabile in $\mathbb{Z}_p^*$. Sia q un divisore primo di $p-1$ e sia α una radice q-esima dell'unità, (ovvero, $\alpha = \alpha_0^{\frac{p-1}{q}i}$, $0 \leq i \leq q-1$, dove α_0 è un elemento primitivo di $\mathbb{Z}_p^*$). Siano $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \mathbb{Z}_q^2$ e sia

$$\mathcal{K} = \left\{(p, q, \alpha, d, \beta) : \beta \equiv \alpha^d \ mod\ p\right\}$$

dove $0 \leq d \leq q-1$. La quadrupla (p, q, α, β) è pubblica, invece d è segreto. Infine, sia $h : \{0,1\}^* \longrightarrow \mathbb{Z}_q$ una funzione hash sicura. Per ogni $K = (p, q, \alpha, d, \beta)$ e un intero segreto k tale che $1 \leq k \leq q-1$ si definisce

$$\text{sig}_K(x, k) = (\gamma, \delta)$$

dove

$$(\gamma, \delta) = (h(x \parallel \alpha^k), (k + d\gamma) \ mod\ q).$$

Per ogni $x \in \{0,1\}^*$ e $\gamma, \delta \in \mathbb{Z}_q$ definiamo

$$\text{ver}_K(x, (\gamma, \delta)) = vero \Longleftrightarrow h(x \parallel \alpha^\delta \beta^{-\gamma}) = \gamma.$$

Proposizione 13.7. Vale che

$$\text{ver}_K(x, (\gamma, \delta)) = vero \Longleftrightarrow h(x \parallel \alpha^\delta \beta^{-\gamma}) = \gamma.$$

Dimostrazione.

($\Rightarrow$) Supponiamo che $\text{ver}_K(x, (\gamma, \delta)) = vero$, dove

$$(\gamma, \delta) = (h(x \parallel \alpha^k), (k + d\gamma) \ mod\ q).$$

Allora $h(x \parallel \alpha^\delta \beta^{-\gamma}) = \gamma$, se e solo se $x \parallel \alpha^\delta \beta^{-\gamma} = x \parallel \alpha^k$ (in binario). Ciò è equivalente a $\alpha^\delta \beta^{-\gamma} = \alpha^k$, ovvero a

$$\begin{aligned} \delta = k + d\gamma &\Longleftrightarrow \alpha^\delta \alpha^{-d\gamma} \equiv \alpha^k \ mod\ q \\ &\Longleftrightarrow \alpha^\delta \beta^{-\gamma} = \alpha^k. \end{aligned} \tag{13.2}$$

Pertanto, $\alpha^\delta \beta^{-\gamma} \equiv \alpha^k \ mod\ p$ essendo $Z_q \leq \mathbb{Z}_p^*$

($\Leftarrow$) Viceversa $h(x \parallel \alpha^\delta \beta^{-\gamma}) = \gamma$. Siccome $\alpha^\delta \beta^{-\gamma} \in Z_q$ allora esiste un intero $1 \leq k \leq q-1$, tale che $\alpha^\delta \beta^{-\gamma} \equiv \alpha^k \ mod\ q$, e quindi $\delta = k + d\gamma$ per (13.2). Pertanto $\text{sig}_K(x, k) = (h(x \parallel \alpha^k), (k + d\gamma) \ mod\ q)$ e quindi

$$\text{ver}_K(x, (h(x \parallel \alpha^k), (k + d\gamma) \ mod\ q)) = vero.$$

□

Caratteristiche del Sistema di Firma di Schnorr

1. Il Sistema di Firma di Schnorr è una modifica del sistema di Firma di ElGamal con il vantaggio che la firma nel primo caso ha lunghezza binaria molto minore che nel secondo. Infatti, allora la firma nel caso di ElGamal è lunga circa $2\log_2 p$ bit, nel caso di Schonorr è $\log_2 q$ bit dove $q \mid p-1$, anche se i calcoli sono fatti in $\mathbb{Z}_p$. In particolare, se $p \simeq 2^{1024}$ e $q \simeq 2^{160}$ allora $2\log_2 p \simeq 2048$ bit, mentre $\log_2 q \simeq 160$.

2. Generalmente nei i sistemi di firma che usano un funzione hash crittografica, la firma è consequenziale all'applicazione della funzione hash. Nel sistema di firma di Schnorr la funzione hash è integrata direttamente nel sistema di firma.

3. La sicurezza del sistema di firma di Schnorr è basata sull'intrattabilità computazionale del PLD in $Z_q \leq \mathbb{Z}_p^*$.

Esempio 13.8. Sia $q = 101$ e $p = 78q+1 = 7879$, allora 3 è un elemento primitivo di $\mathbb{Z}_{7879}^*$. Quindi

$$\alpha = 3^{78} \ mod\ 7879 = 170$$

è una radice 101-esima dell'unità in $\mathbb{Z}_{7879}^*$. Sia $d = 75$, allora

$$\beta = \alpha^d \ mod\ 7879 = 4567.$$

Sia x il messaggio da firmare e sia $k = 50$ l'intero random segreto scelto dall'utente che vuole firmare x. Allora

$$\alpha^k \ mod\ p = 170^{50} \ mod\ 7879 = 2518$$

in binario

$$\alpha^k \ mod\ p = 100111010110.$$

Sia h una qualsiasi funzione hash e supponiamo per semplicità che $h(x \parallel 100111010110) = 96$. Quindi,

$$\delta = (50 + 75 \times 96) \ mod\ 101 = 79$$

allora

$$\mathrm{sig}_{(7879,101,170,75,4567)}(x, 50) = (96, 79).$$

Siccome $170^{79}4567^{-96} \ mod\ 7879 = 2518$, che in binario è 100111010110, e $h(x \parallel 100111010110) = 96$ allora

$$\mathrm{ver}_{(7879,101,170,75,4567)}(x, (96, 79)) = vero.$$

□

13.7.2 L'Algoritmo di Firma Digitale

Definizione 13.9. **(L'algoritmo di Firma Digitale (1994))**
Sia p un primo di L bit dove $L = 64\theta$ dove $8 \leq \theta \leq 16$ tale che il PLD in $\mathbb{Z}_p^*$ sia intrattabile, e siano q un primo di 160 bit che divide $p-1$ e α una radice q-esima dell'unità in $\mathbb{Z}_p^*$. Siano $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \left(\mathbb{Z}_q^*\right)^2$ e

$$\mathcal{K} = \left\{(p, q, \alpha, d, \beta) : \beta \equiv \alpha^d \ mod\ p\right\}$$

dove $0 \leq d \leq q-1$. La quadrupla (p, q, α, β) è pubblica, invece d è segreto. Per ogni $K = (p, q, \alpha, d, \beta)$ e un intero segreto k tale che $1 \leq k \leq q-1$ si definisce

$$\text{sig}_K(x, k) = (\gamma, \delta)$$

dove

$$(\gamma, \delta) = (\alpha^k \ mod\ p \ mod\ q, (\text{SHA}-1(x) + d\gamma)k^{-1} \ mod\ q)$$

(se $\gamma = 0$ o $\delta = 0$, allora viene scelto un nuovo valore di k).
Per ogni $x \in \{0,1\}^*$ e $\gamma, \delta \in \mathbb{Z}_q^*$ vale che

$$\text{ver}_K(x, (\gamma, \delta)) = vero \Longleftrightarrow \alpha^{e_1}\beta^{e_2} \ mod\ p \ mod\ q = \gamma,$$

dove

$$(e_1, e_2) = (\text{SHA}-1(x)\delta^{-1} \ mod\ q, \gamma\delta^{-1} \ mod\ q).$$

Caratteristiche del DSA

1. Il DSA, analogamente al Sistema di Firma di Schnorr, opera in un opportuno sottogruppo Z_q di $\mathbb{Z}_p^*$. Inoltre, le forme delle chiavi nei due sistemi sono uguali.
2. La funzione hash utilizzata nel DSA è la SHA $-$ 1 e la firma consiste di circa 320 bit.
3. La sicurezza del sistema di firma del DSA, come quella di Schnorr, è basata sull'intrattabilità computazionale del PLD in $Z_q \leq \mathbb{Z}_p^*$.
4. Si noti che se un utente ottiene $\delta \equiv 0 \ mod\ q$, dovrebbe eliminarlo e scegliere un nuovo intero random. La probabilità che questo accada è $1/q \simeq 2^{-160}$.

Esempio 13.10. Siano $(p, q, \alpha, d, \beta) = (7879, 101, 170, 75, 4567)$ e $k = 50$ come nell'**Esempio 13.8** e supponiamo che l'utente voglia firmare $\text{SHA}-1(x) = 22$. Allora $k^{-1} \ mod\ 101 = 99$ e

$$\begin{aligned} \gamma &= (170^{50} \ mod\ 7879) \ mod\ 101 = 2518 \ mod\ 101 = 94 \\ \delta &= ((22 + 75 \times 94) \times 99) \ mod\ 101 = 97. \end{aligned}$$

Quindi

$$\text{sig}_{(7879,101,170,75,4567)}(x,50) = (94,97).$$

Allora

$$\begin{aligned} \delta^{-1} &= 97^{-1} \ mod\ 101 = 25 \\ e_1 &= (22 \times 25) \ mod\ 101 = 45 \\ e_2 &= (94 \times 25) \ mod\ 101 = 27 \end{aligned}$$

Siccome

$$\left(170^{45}4567^{27} \ mod\ 7879\right) \ mod\ 101 = 2528 \ mod\ 101 = 94,$$

allora

$$\text{ver}_{(7879,101,170,75,4567)}(x,(94,97)) = vero.$$

□

13.7.3 L'Algoritmo di Firma Digitale basato sulle curve ellittiche (ECDSA)

Definizione 13.11. **(Algoritmo di Firma Digitale basato sulle curve ellittiche (2000))**
Sia $\mathcal{E}$ una curve ellittica su $GF(p)$ dove p è un primo oppure una potenza di 2. Sia Q un punto di $\mathcal{E}$ di ordine un primo q tale che IL PLD sia intrattabile in $\langle Q \rangle$. Siano $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \left(\mathbb{Z}_q^*\right)^2$ e

$$\mathcal{K} = \{(p,q,\mathcal{E},Q,m,B) : B = mQ\}$$

dove $0 \leq m \leq q-1$. La quintupla $(p,q,\mathcal{E},Q,B)$ è pubblica, invece m è segreto. Per ogni $K = (p,q,\mathcal{E},Q,m,B)$ e un intero segreto random k tale che $1 \leq k \leq q-1$ si definisce

$$\text{sig}_K(x,k) = (r,s)$$

dove

$$\begin{cases} kQ = (u,v) \\ r = u \ mod\ q \\ s = k^{-1}(\text{SHA}-1(x) + mr) \ mod\ q \end{cases}$$

(se $r = 0$ o $s = 0$, allora viene scelto un nuovo valore di k).
Per ogni $x \in \{0,1\}^*$ e $r,s \in \mathbb{Z}_q^*$ vale che

$$\begin{cases} w = s^{-1} \ mod\ q \\ i = w\,(\text{SHA}-1(x)) \ mod\ q \\ j = wr \ mod\ q \\ (u,v) = iQ + jB \end{cases}$$

allora

$$\text{ver}_K(x,(r,s)) = vero \iff u \ mod\ q = r.$$

Esempio 13.12. Si consideri la curva ellittica $\mathcal{E} : y^2 = x^3 + x + 6$ a coefficienti in $GF(11)$. Sia

$$K = (11, 13, (2, 7), 7, (7, 2))$$

e supponiamo che $\text{SHA} - 1(x) = 4$ e che il generico utente voglia firmare x usando l'intero casuale $k = 3$. Allora

$$\begin{aligned} (u, v) &= 3(2, 7) = (8, 3) \\ r &= 8 \ mod \ 13 = 8 \\ s &= 3^{-1}(4 + 7 \times 8) \ mod \ 13 = 7. \end{aligned}$$

Quindi,

$$\text{sig}_{(11,13,(2,7),7,(7,2))}(x, 3) = (8, 7).$$

Il destinatario del messaggio firmato calcola

$$\begin{cases} w = 7^{-1} \ mod \ 13 = 2 \\ i = (2 \times 4) \ mod \ 13 = 8 \\ j = (2 \times 8) \ mod \ 13 = 3 \\ (u, v) = 8(2, 7) + 3(7, 2) = (8, 3) \end{cases}$$

Siccome $u \ mod \ 13 = 8 = r$, allora

$$\text{ver}_{(11,13,(2,7),7,(7,2))}(x, (8, 7)) = vero$$

dimostra la certezza del mittente.

□

13.8 Il Sistema di Firma di Lamport

Il seguente sistema di firma, dovuto a Lamport, fonda la sua sicurezza su una funzione one-way (cioè è dimostrabilmente sicuro) ed ogni chiave è utilizzata per firmare un unico documento.

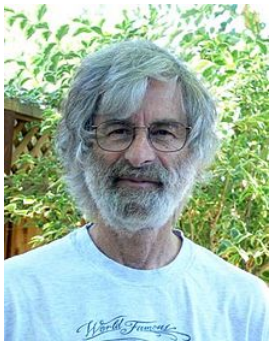


Figura 13.2: Leslie Lamport (1941)

Definizione 13.13. **(Sistema di Firma di Lamport (1976))**
Sia $k \in \mathbb{N}$ e siano $\mathcal{P} = \{0,1\}^k$, $f : Y \longrightarrow Z$ una funzione (pubblica) one-way, $\mathcal{A} = Y^k$ e $\mathcal{K} \subseteq Y^{2k} \times Z^{2k}$. Sia $y_{ij} \in Y$ scelto in modo random, dove $1 \leq i \leq k$ e $j = 0, 1$, e sia $z_{ij} = f(y_{ij})$. Sia

$$K = (y_{10}, ..., y_{k0}, y_{11}, ..., y_{k1}, z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1}),$$

dove

$$(y_{10}, ..., y_{k0}, y_{11}, ..., y_{k1})$$

è privata, invece

$$(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1})$$

è pubblica. Allora definiamo

$$\text{sig}_K(x_1, ..., x_k) = (y_{1,x_1}, ..., y_{k,x_k}).$$

Una firma $(a_1, ..., a_k)$ su un messaggio $(x_1, ..., x_k)$ è verificata come segue

$$\text{ver}_K((x_1, ..., x_k), (a_1, ..., a_k)) = vero \Longleftrightarrow f(a_i) = z_{i,x_i},\ 1 \leq i \leq k.$$

Remark 13.14. Il sistema di Firma di Lamport può essere costruito a partire da dalla funzione di esponenziazione modulare, che si crede essere one-way: Se p è un primo e α è un elemento primitivo di $\mathbb{Z}_p^*$, allora

$$f : \{0, ..., p-2\} \longrightarrow \mathbb{Z}_p^*, x \longmapsto \alpha^x \ \ mod\ p.$$

Esempio 13.15. Sia $p = 7879$ e $\alpha = 3$ un elemento primitivo di $\mathbb{Z}^*_{7879}$ e sia $k = 3$.

$$f : \{0, ..., 7877\} \longrightarrow \mathbb{Z}^*_{7879}, x \longmapsto 3^x \ \ mod\ 7879.$$

Sia

$$(y_{10}, y_{20}, y_{30}, y_{11}, y_{21}, y_{31}) = (5831, 803, 4285, 735, 2467, 6449)$$

e le rispettive immagini mediante f

$$(z_{10}, z_{20}, z_{30}, z_{11}, z_{21}, z_{31}) = (2009, 4672, 268, 3810, 4721, 5731).$$

Quindi la chiave è

$$\begin{aligned} K = (&5831, 803, 4285, 735, 2467, 6449, \\ &2009, 4672, 268, 3810, 4721, 5731). \end{aligned} \tag{13.3}$$

Supponiamo che l'utente voglia firmare il messaggio $x = (1, 1, 0)$, allora

$$\text{sig}_K(1, 1, 0) = (y_{11}, y_{21}, y_{30}) = (735, 2467, 4285).$$

Il destinatario prestabilito una volta ricevuto il messaggio firmato $((1, 1, 0), (735, 2467, 4285))$ calcola

$$\begin{cases} f(375) = 3^{375} \ \ mod\ 7879 = 3810 \\ f(2467) = 3^{2467} \ \ mod\ 7879 = 4721 \\ f(4285) = 3^{4285} \ \ mod\ 7879 = 268 \end{cases}$$

e verifica in (13.3) che $(3810, 4721, 268) = (z_{11}, z_{21}, z_{30})$. In tal caso,

$$\text{ver}_K((1, 0, 0), (735, 2467, 4285)) = vero.$$

□

1. Se la chiave è utilizzata per firmare più di un messaggio, il sistema di firma di Lamport non è sicuro.

   Infatti, se per esempio $k = 3$ e quindi i seguenti messaggi $x = (0, 1, 1)$ e $x' = (1, 0, 1)$ sono firmati con la stessa chiave

   $$K = (y_{10}, y_{20}, y_{30}, y_{11}, y_{21}, y_{31}, z_{10}, z_{20}, z_{30}, z_{11}, z_{21}, z_{31})$$

   segreta, allora un avversario produce una $(2, 1)$-falsificazione esistenziale. Infatti, se le firma di x è (y_{10}, y_{21}, y_{31}) e quella di x' è (y_{11}, y_{20}, y_{31}), allora l'avversario firma $x'' = (1, 1, 1)$ con (y_{11}, y_{21}, y_{31}) e $x''' = (0, 0, 1)$ con (y_{10}, y_{20}, y_{31}). Entrambe le firme sono valide!

2. Proviamo che il sistema di firma di Lamport è dimostrabilemente sicuro nella modalità **one-time**, cioè la chiave è utilizzata per firmare un solo documento. In particolare, proviamo che sistema è sicuro rispetto all'attacco basato sulla conoscenza della chiave pubblica, se f è biettiva e la chiave pubblica $(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1})$ è costituita da $2k$ elementi distinti.

   Sia Lamport-Falsificazione un algoritmo deterministico che fornita una chiave pubblica $(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1})$ restituisce in output la coppie di k-ple $((x_1, ..., x_k), (a_1, ..., a_k))$ dove $f(a_i) = z_{i,x_i}$, $1 \leq i \leq k$. e consideriamo il seguente algoritmo

Algoritmo 13.16. **(Lamport-immagine inversa(z))**

esterni: f, Lamport-Falsificazione

comment: supponiamo che $f : Y \longrightarrow Z$ sia una biiezione

fissati $i_0 \in \{1, ...k\}$ e $j_0 \in \{0, 1\}$ costruiamo una chiave pubblica

$(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1})$ tale che $z_{i_0 j_0} = z$.

$((x_1, ..., x_k), (a_1, ..., a_k)) \leftarrow$ Lamport-Falsificazione$((z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1}))$

if $x_{i_0} = j_0$

then return (a_{i_0})

else return (insuccesso)

Noi stiamo assumendo che l'algoritmo Lamport-Falsificazione trovi sempre una falsificazione $((x_1, ..., x_k), (a_1, ..., a_k))$ relativa alla chiave pubblica $(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1})$. Se vale $x_{i_0} = j_0$, allora $f(a_{i_0}) = z_{i_0, x_{i_0}} = z_{i_0, j_0} = z$ e quindi $a_{i_0} \in f^{-1}(z)$, cioè abbiamo trovato un elemento dell'immagine inversa di z. Vogliamo provare nel seguente teorema che la probabilità media di successo nel determinare un elemento dell'immagine inversa di $f^{-1}(z)$, al variare di z in Z è almeno $1/2$.

Teorema 13.17. *Sia $f : Y \longrightarrow Z$ una funzione one-way biiettiva, e supponiamo che esista un algoritmo deterministico Lamport-Falsificazione* (LF) *che produce una falsificazione esistenziale basata sulla sola conoscenza della chiave pubblica nel Sistema di Firma di Lamport per ogni chiave $(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1}) \in Z^{2k}$ costituita da $2k$ termini distinti. Allora esiste un algoritmo Lamport-immagine inversa* (LII) *che determina immagini inverse di elementi random in Z con probabilità maggiore o uguale a $1/2$.*

Dimostrazione. Sia $\mathcal{S} \subseteq Z^{2k}$ l'insieme di tutte le possibili chiavi pubbliche, e per ogni $z \in Z$ sia S_z l'insieme

$$\{(z_{10}, ..., z_{k0}, z_{11}, ..., z_{k1}) \in \mathcal{S} : \exists i \in \{1, ...k\}, j \in \{1, 2\} \text{ t.c. } z_{ij} = z\}.$$

Denotata con $\mathcal{Z}$ la generica chiave pubblica, allora $\mathcal{T}_z$ è l'insieme degli $\mathcal{Z} \in \mathcal{S}_z$ tale che Lamport-immagine inversa(z) successo, se $\mathcal{Z}$ è scelta come chiave pubblica nell'algoritmo Lamport-immagine inversa, essendo questo randomizzato. Posto $s = |\mathcal{S}|$, $s_z = |\mathcal{S}_z|$ e $t_z = |\mathcal{T}_z|$, contiamo in due modi diversi

$$\sum_{\substack{z \in Z, \mathcal{Z} \in \mathcal{S} \\ \mathcal{Z} \in \mathcal{T}_z}} (\mathcal{Z}, z). \tag{13.4}$$

Fissato $z \in Z$ il numero degli $\mathcal{Z} \in \mathcal{T}_z$ è t_z quindi (13.4) vale $\sum_{z \in Z} t_z$. D'altra parte, fissato $\mathcal{Z} \in \mathcal{S}$ l'algoritmo deterministico Lamport-Falsificazione individua esattamente gli inversi tramite f di k elementi di Z. Quindi esistono esattamente k elementi $z_1, ..., z_k$ di Z tali che $\mathcal{Z} \in \mathcal{T}_{z_j}$, $j = 1, ..., k$. Pertanto, (13.4) vale $\sum_{\mathcal{Z} \in \mathcal{S}} k = ks$. Quindi, vale che

$$\sum_{z \in Z} t_z = ks \tag{13.5}$$

Ora contiamo in due modi diversi

$$\sum_{\substack{z \in Z, \mathcal{Z} \in \mathcal{S} \\ \mathcal{Z} \in \mathcal{S}_z}} (\mathcal{Z}, z). \tag{13.6}$$

Fissato $z \in Z$ il numero degli $\mathcal{Z} \in \mathcal{S}_z$ è s_z quindi (13.4) vale $\sum_{z \in Z} s_z$. D'altra parte, per ipotesi, fissato $\mathcal{Z} \in \mathcal{S}$ il numero degli z che $\mathcal{Z}$ contiene è $2k$ elementi di Z. Quindi, $\sum_{\mathcal{Z} \in \mathcal{S}} 2k = 2ks$. Pertanto si ha

$$\sum_{z \in Z} s_z = 2ks$$

Sia z_0 tale che $s_{z_0} = \max\{s_z : z \in Z\}$. Allora

$$2ks \leq s_{z_0} |Z|$$

Sia p_z la probabilità che l'algoritmo Lamport-immagine inversa(z) abbia successo. Chiaramente, $p_z = t_z / s_z$. Allora,

$$\bar{p} = \frac{1}{|Z|} \sum_{z \in Z} p_z = \frac{1}{|Z|} \sum_{z \in Z} \frac{t_z}{s_z} \geq \frac{1}{|Z| \, s_{z_0}} \sum_{z \in Z} t_z = \frac{ks}{2ks} = \frac{1}{2}.$$

Pertanto, $\bar{p} \geq 1/2$ che è l'asserto.

□

Remark 13.18. Il sistema di Firma di Lamport, sebbene elegante, risulta poco pratico per via della grandezza delle firme che esso produce. Supponiamo per esempio di prendere come f l'esponenziazione modulare (quella utilizzata per il crittosistema RSA). Cioè se Se p è un primo e α è un elemento primitivo di $\mathbb{Z}_p^*$, allora

$$f : \{0, ..., p-2\} \longrightarrow \mathbb{Z}_p^*, x \longmapsto \alpha^x \ \ mod\ p.$$

Per essere sicura $p \simeq 1024$ bit e quindi la firma nel sistema di Lamport deve essere lungo $1024 \times k$ bit.

13.9 Firme non ripudiabili

Le firme non ripudiabili sono state introdotte da **Chaum** e **Van Anterwerpen** nel 1989 e presentano diverse novità:

1. La firma non può essere verificata senza la cooperazione del firmatario: questo protegge il firmatario contro la possibilità che i documenti firmati siano duplicati e distribuiti elettronicamente. La verifica è compiuta attraverso il **protocollo challenge and repsonse**.

2. La firma non può essere ripudiata: il firmatario non può asserire che una firma valida sia stata falsificata o possa fare in modo che la firma non sia verificata. Per prevenire questo tipo avvenimenti, le firme non ripudiabili contengono un **protocollo di disconoscimento** attraverso il quale il firmatario può provare che una firma falsificata è di fatto una falsificazione. (Se il firmatario si rifiuta di prendere parte nel protocollo di disconoscimento ciò viene visto come una prova della validità di una firma).

Figura 13.3: David Lee Chaum (1955)

Come vedremo, i sistemi di firma non ripudiabili consistono di tre componenti:

- Algoritmo di firma.
- Algoritmo di verifica.
- Algoritmo di disconoscimento.

Definizione 13.19. **(Sistema di Firma di Chaum - Van Anterwerpen (1989))**
Sia $p = 2q + 1$ un primo tale che q sia un primo e che il PLD sia intrattabile in $\mathbb{Z}_p^*$. Siano $\alpha \in \mathbb{Z}_p^*$ un elemento di ordine q, $1 \leq m \leq q-1$ e $\beta \equiv \alpha^m \ mod\ p$ e sia $G \cong Z_q \leq \mathbb{Z}_p^*$ (G è il sottogruppo dei quadrati di $\mathbb{Z}_p^*$). Siano $\mathcal{P} = \mathcal{A} = G$ e sia

$$\mathcal{K} = \{(p, \alpha, m, \beta) : \beta \equiv \alpha^m \ mod\ p\}$$

La terna (p, α, β) è la chiave pubblica, m è la chiave privata.
Se $K = (p, \alpha, m, \beta)$ e $x \in G$ si definisca

$$y = \text{sig}_K(x) = x^m \ mod\ p.$$

Per $x, y \in G$ la verifica della firma avviene secondo il seguente protocollo:

1. L'utente B destinatario del documento firmato sceglie in modo random $e_1, e_2 \in \mathbb{Z}_q^*$.
2. L'utente B calcola $c \equiv y^{e_1}\beta^{e_2} \ mod\ p$ e lo trasmette al firmatario A (**challenge**).
3. L'utente A calcola $d \equiv c^{m^{-1} \ mod\ q} \ modp$ e lo trasmette a B (**response**).
4. L'utente B accetta y come firma valida se, e solo se,
$$d \equiv x^{e_1}\alpha^{e_2} \ mod\ p.$$

Proposizione 13.20. Siano $x, y \in G$, $\beta \equiv \alpha^m \ mod\ p$, $c \equiv y^{e_1}\beta^{e_2} \ mod\ p$, $d \equiv c^{m^{-1}} \ mod\ p$, allora

$$y \equiv x^m \ mod\ p \iff d \equiv x^{e_1}\alpha^{e_2} \ mod\ p.$$

Dimostrazione. Da $d \equiv c^{m^{-1} \ mod\ q} \ mod\ p$ e $c \equiv y^{e_1}\beta^{e_2} \ mod\ p$ segue che $d \equiv y^{e_1 m^{-1} \ mod\ q}\beta^{e_2 m^{-1} \ mod\ q} \ mod\ p$ e quindi $d \equiv y^{e_1 m^{-1} \ mod\ q}\alpha^{e_2} \ mod\ p$ essendo $\beta \equiv \alpha^m \ mod\ p$.
Pertanto,

$$\begin{aligned} d \equiv x^{e_1}\alpha^{e_2} \ mod\ p &\iff x^{e_1}\alpha^{e_2} \equiv y^{e_1 m^{-1} \ mod\ q}\alpha^{e_2} \ mod\ p \\ &\iff x^{e_1} \equiv y^{e_1 m^{-1} \ mod\ q} \ mod\ p \end{aligned} \tag{13.7}$$

Siccome $e_1 \in \mathbb{Z}_q$, allora esiste v_1 inverso di e_1 in G. Allora $e_1 v_1 = \theta q + 1$. Pertanto

$$\begin{aligned} x^{e_1} \equiv y^{e_1 m^{-1} \ mod\ q} \ mod\ p &\iff x^{e_1 v_1} \equiv y^{e_1 v_1 m^{-1} \ mod\ q} \ mod\ p \\ &\iff x^{\theta q+1} \equiv y^{(\theta q+1)m^{-1} \ mod\ q} \ mod\ p \\ &\iff x \equiv y^{m^{-1} \ mod\ q} \ mod\ p \\ &\iff y \equiv x^m \ mod\ p. \end{aligned} \tag{13.8}$$

Pertanto, da (13.7) e (13.8) segue l'asserto.

□

Esempio 13.21. Sia $p = 467 = 2 \times 233 + 1$ un primo e sia $\alpha_0 = 2$ un elemento primitivo di $\mathbb{Z}^*_{467}$. Allora $\alpha = \alpha_0^2 = 4$ è un elemento primitivo del sottogruppo $G \cong Z_{233}$ dei quadrati di $\mathbb{Z}^*_{467}$. Sia $m = 101$, allora $\beta \equiv 4^{101} \ mod\ 467 = 449$.

Un utente A vuole firmare un messaggio $x = 119$, allora

$$y = \text{sig}_{(467,4,101,449)}119 = 119^{101} \ mod\ 467 = 129$$

e trasmette il messaggio firmato $(x, y) = (119, 129)$ all'utente B. Una volta ricevuto $(119, 129)$ l'utente B, vuole verificare la firma. Allora B opera come segue

1. sceglie due elementi $e_1 = 38$ e $e_2 = 397$ in $\mathbb{Z}_q$
2. calcola $c = 129^{38}449^{397} \ mod\ 467 = 13$ e lo trasmette ad A
3. A calcola $d = 13^{101^{-1} \ mod\ 233} \ mod\ 467 = 13^{30} \ mod\ 467 = 9$ e lo ritrasmette a B.
4. B calcola $109^{38}4^{397} \ mod\ 467 = 9$ che è uguale a $d = 9$ trasmesso da A e quindi è sicuro che la firma è valida, cioè

$$\text{ver}_{(467,4,101,449)}(119, 129) = vero.$$

□

Adesso proviamo che è molto bassa la probailità che un utente A inganni un utente B facendo accettare come valida una firma falsa. Il seguente risultato è indipendente dalle risorse computazionali dell'avversario (l'utente A), cioè la sicurezza è incondizionata.

Proposizione 13.22. Valgono i seguenti fatti:

1. Se vale $c \equiv y^{\bar{e}_1}\beta^{\bar{e}_2} \ mod\ p$, allora q è il numero delle coppie $(e_1, e_2) \in \mathbb{Z}_q^2$ tale che $c \equiv y^{e_1}\beta^{e_2} \ mod\ p$.
2. Se vale **(1)** e $y \not\equiv x^m \ mod\ p$, allora esiste un'unica coppia (e_1, e_2) (tra le q coppie) tale che

$$\begin{cases} c \equiv y^{e_1}\beta^{e_2} \ mod\ p \\ d \equiv x^{e_1}\alpha^{e_2} \ mod\ p \end{cases} \tag{13.9}$$

Dimostrazione. Sia $(\bar{e}_1, \bar{e}_2)$ una coppia tale $c \equiv y^{\bar{e}_1}\beta^{\bar{e}_2} \ mod\ p$. Siccome G è ciclico di ordine primo q esiste un intero k tale che $y \equiv \beta^k \ mod\ q$. Quindi, $y \equiv \beta^k \ mod\ p$, essendo $G \leq \mathbb{Z}^*_p$. Pertanto, $c \equiv \beta^{k\bar{e}_1+\bar{e}_2} \ mod\ p$, allora $(e_1, k\bar{e}_1 + \bar{e}_2 - ke_1)$ è tale che $c \equiv y^{e_1}\beta^{k\bar{e}_1+\bar{e}_2-ke_1} \ mod\ p$. Pertanto, fissata una coppia esistono esattamente q coppie (e_1, e_2) tali che $y^{e_1}\beta^{e_2} \ mod\ p \equiv y^{\bar{e}_1}\beta^{\bar{e}_2} \ mod\ p \equiv c$, che è l'asserto **(1)**.

Supponiamo che valga **(1)** e $y \notin x^m \mod p$, poiché α è un generatore di G, allora esistono $0 \leq i, j, k, \ell \leq q-1$ tale che $c = \alpha^i$, $d = \alpha^j$, $x = \alpha^k$ e $y = \alpha^\ell$. Siccome $G \leq \mathbb{Z}_p^*$, le precedenti uguaglianze sono da intendersi modulo p. Quindi

$$\begin{cases} c \equiv y^{e_1}\beta^{e_2} \mod p \\ d \equiv x^{e_1}\alpha^{e_2} \mod p \end{cases} \iff \begin{cases} \alpha^i \equiv \alpha^{\ell e_1 + m e_2} \mod p \\ \alpha^j \equiv \alpha^{k e_1 + e_2} \mod p \end{cases}$$

che è equivalente a

$$\begin{cases} i \equiv \ell e_1 + m e_2 \mod q \\ j \equiv k e_1 + e_2 \mod q \end{cases} \tag{13.10}$$

siccome $o(\alpha) = q$. Il sistema è lineare nelle indeterminate e_1, e_2 e a coefficienti nel campo $\mathbb{Z}_q$, essendo q primo. Siccome $y \notin x^m \mod p$ e poichè $G \leq \mathbb{Z}_p^*$, allora $\alpha^\ell \notin \alpha^{km} \mod p$ e quindi $\ell - km \not\equiv 0 \mod q$, ovvero

$$\det \begin{pmatrix} \ell & m \\ k & 1 \end{pmatrix} = \ell - km \not\equiv 0 \mod q,$$

allora il sistema (13.10) ammette un'unica soluzione. Pertanto, fissato d esiste un unica coppia (e_1, e_2) tale che valga

$$\begin{cases} c \equiv y^{e_1}\beta^{e_2} \mod p \\ d \equiv x^{e_1}\alpha^{e_2} \mod p. \end{cases}$$

□

Teorema 13.23. *Se $y \notin x^m \mod p$, allora la probabilità che l'utente B accetti y come firma valida per il messaggio x è $1/q$.*

Dimostrazione. Supponiamo che A inganni B trasmettendo la coppia (x, y) con $y \notin x^m \mod p$ firma non valida. Allora, nel protocollo challenge and response vale che:

1. L'utente B sceglie in modo random $\bar{e}_1, \bar{e}_2 \in \mathbb{Z}_q$.
2. L'utente B calcola $c \equiv y^{\bar{e}_1}\beta^{\bar{e}_2} \mod p$ e lo trasmette ad A.
3. L'utente A **sceglie** $\bar{d}$ e lo trasmette a B.
4. L'utente B accetta y come firma valida se, e solo se,
$$\bar{d} \equiv x^{e_1}\alpha^{e_2} \mod p.$$

Quindi la probabilità che B accetti y come firma valida è uguale alla probabilità che $(\bar{e}_1, \bar{e}_2)$ scelti da B coincidano con la soluzione del sistema (13.9)

$$\begin{cases} c \equiv y^{e_1}\beta^{e_2} \mod p \\ \bar{d} \equiv x^{e_1}\alpha^{e_2} \mod p. \end{cases}$$

che per la **Proposizione 13.22** è $1/q$.

□

Analizziamo ora il Protocollo di disconoscimento che consiste nell'applicare il protocollo di verifica due volte.

Algoritmo 13.24. **(Disconoscimento)**

1. L'utente B sceglie in modo random $e_1, e_2 \in \mathbb{Z}_q^*$.
2. L'utente B calcola $c = y^{e_1}\beta^{e_2} \mod p$ e lo trasmette al firmatario A.
3. L'utente A calcola $d = c^{m^{-1} \mod q} \mod p$ e lo trasmette a B.
4. L'utente B verifica $d \not\equiv x^{e_1}\alpha^{e_2} \mod p$.
5. L'utente B sceglie in modo random $f_1, f_2 \in \mathbb{Z}_q^*$.
6. L'utente B calcola $C = y^{f_1}\beta^{f_2} \mod p$ e lo trasmette al firmatario A.
7. L'utente A calcola $D = C^{m^{-1} \mod q} \mod p$ e lo trasmette a B.
8. L'utente B verifica $D \not\equiv x^{f_1}\alpha^{f_2} \mod p$.
9. L'utente B conclude che y è una falsificazione se, e solo se, $(d\alpha^{-e_2})^{f_1} \equiv \left(D\alpha^{-f_2}\right)^{e_1} \mod p$.

Sia negli step **1**–**4** e che in quelli **5**–**8** si applica l'algoritmo di verifica, lo step **9** rappresenta una verifica di omogeneità e premette all'utente B di capire se l'utente A produce le risposte nella maniera specificata dal protocollo.

Proviamo i seguenti fatti:

1. L'utente A è in grado di convincere l'utente B che una firma non valida è una falsificazione.
2. La probabilità che l'utente A sia in grado di convincere l'utente B che una firma valida è una falsificazione è molto bassa.

Teorema 13.25. *Se $y \not\equiv x^m \mod p$ e gli utenti A e B seguono il protocollo di disconoscimento, allora*

$$(d\alpha^{-e_2})^{f_1} \equiv \left(D\alpha^{-f_2}\right)^{e_1} \mod p.$$

Dimostrazione. Siccome vale che

$$\begin{cases} d \equiv c^{m^{-1} \mod q} \mod p \\ c \equiv y^{e_1}\beta^{e_2} \mod p \\ \beta \equiv \alpha^m \mod p \end{cases}$$

allora

$$\begin{aligned}(d\alpha^{-e_2})^{f_1} &\equiv \left((y^{e_1}\beta^{e_2})^{m^{-1} \bmod q}\alpha^{-e_2}\right)^{f_1} \bmod p \\ &\equiv y^{e_1 f_1 (m^{-1} \bmod q)}\beta^{e_2 f_1 (m^{-1} \bmod q)}\alpha^{-e_2 f_1} \bmod p \\ &\equiv y^{e_1 f_1 (m^{-1} \bmod q)}\alpha^{e_2 f_1}\alpha^{-e_2 f_1} \bmod p \\ &\equiv y^{e_1 f_1 (m^{-1} \bmod q)} \bmod p.\end{aligned}$$

quindi,

$$(d\alpha^{-e_2})^{f_1} \equiv y^{e_1 f_1 (m^{-1} \bmod q)} \bmod p. \tag{13.11}$$

Analogamente,

$$\begin{cases} D \equiv C^{m^{-1} \bmod q} \bmod p \\ C \equiv y^{f_1}\beta^{f_2} \bmod p \\ \beta \equiv \alpha^m \bmod p \end{cases}$$

implica

$$\left(D\alpha^{-f_2}\right)^{e_1} \equiv y^{e_1 f_1 (m^{-1} \bmod q)} \bmod p. \tag{13.12}$$

Da (13.11) e (13.12), segue la tesi.

□

Esempio 13.26. Sia $p = 467$, $\alpha = 4$ un generatore di G il sottogruppo dei quadrati di $\mathbb{Z}^*_{467}$, $m = 101$ e $\beta = 449$. Supponiamo che l'utente A voglia ingannare l'utente B. Sia $y = 83$ la firma **fasulla** dell'utente A al messaggio $x = 286$.

1. L'utente B sceglie $e_1 = 45$ e $e_2 = 237$ in $\mathbb{Z}^*_{233}$.
2. L'utente B calcola $c = 83^{45}449^{237} \bmod 467 = 305$ e lo trasmette ad A.
3. L'utente A calcola $d = 305^{101^{-1} \bmod 233} \bmod 467 = 109$ e lo trasmette a B.
4. L'utente B calcola $286^{45}4^{237} \bmod 467 = 149 \neq 109$.
5. L'utente B sceglie $f_1 = 125$ e $f_2 = 9$ in $\mathbb{Z}^*_{233}$.
6. L'utente B calcola $C = 83^{125}449^{9} \bmod 467 = 270$ e lo trasmette ad A.
7. L'utente A calcola $D = 270^{101^{-1} \bmod 233} \bmod 467 = 68$ e lo trasmette a B.
8. L'utente B calcola $286^{125}4^{9} \bmod 467 = 25 \neq 68$.
9. L'utente B calcola $\left(109 \times 4^{-237}\right)^{125} \bmod 467 = 188$ e $\left(68 \times 4^{-9}\right)^{45} \bmod 467 = 188$ e si convince che la firma dell'utente A è falsa.

□

Analizziamo il caso in cui l'utente A ripudi una firma valida, ovvero $y \equiv x^m \bmod p$, e quindi produca d e D **non** seguendo il protocollo, cioè tali che

$$\begin{cases} d \not\equiv x^{e_1}\alpha^{e_2} \bmod p \\ D \not\equiv x^{f_1}\alpha^{f_2} \bmod p \\ (d\alpha^{-e_2})^{f_1} \not\equiv \left(D\alpha^{-f_2}\right)^{e_1} \bmod p. \end{cases} \tag{13.13}$$

Teorema 13.27. *Supponiamo che $y \equiv x^m \ mod\ p$ e che l'utente B segua il protocollo di disconoscimento. Se $d \not\equiv x^{e_1}\alpha^{e_2} \ mod\ p$ e $D \not\equiv x^{f_1}\alpha^{f_2} \ mod\ p$, allora*

$$\mathbf{P}\left[D\alpha^{-f_2} \equiv (d\alpha^{-e_2})^{f_1/e_1} \ mod\ p\right] = 1/q.$$

Dimostrazione. Sia E_1 l'evento $(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \ mod\ p$. Allora $D\alpha^{-f_2} \equiv (d\alpha^{-e_2})^{f_1/e_1} \ mod\ p$ e quindi

$$D \equiv {d_0}^{f_1}\alpha^{f_2} \ mod\ p \text{ e } d_0 = d^{1/e_1}\alpha^{-e_2/e_1}.$$

Se E_2 denota l'evento $y \equiv d_0^m \ mod\ p$, proviamo che $\mathbf{P}[E_1 \cap E_2] = 0$. Infatti, se $\mathbf{P}[E_1 \cap E_2] > 0$ allora vale che

$$\begin{cases} y \equiv x^m \ mod\ p \\ y \equiv d_0^m \ mod\ p \end{cases}$$

e quindi

$$x^m \equiv d_0^m \ mod\ p,$$

allora esistono $0 \le i, j \le q-1$ tali che $x = \alpha^i$ e $d_0 = \alpha^j$. Pertanto, $\alpha^{im} \equiv \alpha^{jm} \ mod\ p$ e quindi $\alpha^{im} = \alpha^{jm}$ in $G \le \mathbb{Z}_p^*$ da cui segue che $q \mid m(i-j)$. Siccome $1 \le m \le q-1$ e q è primo, allora $q \mid (i-j)$ da cui segue che

$$x = \alpha^i = \alpha^j = d_0.$$

Siccome $d \not\equiv x^{e_1}\alpha^{e_2} \ mod\ p$ allora $x \not\equiv d^{1/e_1}\alpha^{-e_2/e_1} \ mod\ p$ e quindi $d_0 \not\equiv d^{1/e_1}\alpha^{-e_2/e_1} \ mod\ p$, ma ciò è assurdo essendo $d_0 = d^{1/e_1}\alpha^{-e_2/e_1}$. Pertanto, $\mathbf{P}[E_1 \cap E_2] = 0$ e quindi,

$$\mathbf{P}[E_1] = \mathbf{P}[E_1 \cap E_2] + \mathbf{P}[E_1 \cap E_2^c] = 0 + \mathbf{P}[E_1 \cap E_2^c] = \mathbf{P}[E_1 \cap E_2^c].$$

Dal **Teorema 13.23** segue che

$$\mathbf{P}[E_1] = \mathbf{P}[D\alpha^{-f_2} \equiv (d\alpha^{-e_2})^{f_1/e_1} \ mod\ p, y \not\equiv d_0^m \ mod\ p] = 1/q.$$

□

13.10 Firme fail-stop

Un **sistema di firma fail-stop** fornisce ulteriore sicurezza per prevenire la possibilità che un avversario avente infinite risorse computazionali sia in grado di falsificare una firma. Quando ciò si verifica, l'utente a cui è stata falsificata la firma è in grado di dimostrare l'avvenuta falsificazione.

In questo paragrafo finale, descriviamo il sistema di firma fail-stop costruito da **Pedersen-van Heyst** nel 1992.

Figura 13.4: Torben Pryds Pedersen e Pater Eugène van Heyst (1960)

Il sistema, come quello di Lamport è one-time e consiste di tre algoritmi:

- l'algoritmo di firma.
- l'algoritmo di verifica.
- l'algoritmo che dimostra l'avvenuta falsificazione.

Definizione 13.28. **(Sistema di Firma di Pedersen-van Heyst (1992))** Sia $p = 2q + 1$ un primo tale che q sia primo e che il PLD sia intrattabile in $\mathbb{Z}_p^*$. Siano α un elemento di ordine q in $\mathbb{Z}_p^*$, $1 \leq a_0 \leq q - 1$ e $\beta \equiv \alpha^{a_0} \ mod\ p$. La quadrupla $(p, q, \alpha, a_0, \beta)$ sono scelti da un ente di fiducia: (p, q, α, β) è pubblico, invece a_0 è privato (anche agli utenti). Siano $\mathcal{P} = \mathbb{Z}_q$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$ e la generica chiave è la sestupla

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2),$$

dove $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$,

$$\begin{cases} \gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ mod\ p \\ \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ mod\ p \end{cases} \tag{13.14}$$

con (γ_1, γ_2) pubblico e (a_1, a_2, b_1, b_2) privato.
Se $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ e x è un messaggio, allora

$$\text{sig}_K(x) = (a_1 + xb_1 \ mod\ q, a_2 + xb_2 \ mod\ q).$$

Se $(y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$, allora

$$\text{ver}_K(x, y) = vero \iff \gamma_1\gamma_2^x \equiv \alpha^{y_1}\beta^{y_2} \ mod\ p.$$

Proposizione 13.29. Vale che

$$(y_1, y_2) = ((a_1 + xb_1) \ mod\ q, (a_2 + xb_2) \ mod\ q) \Rightarrow \gamma_1\gamma_2^x \equiv \alpha^{y_1}\beta^{y_2} mod\ p$$

Dimostrazione. Per (13.14) risulta

$$\begin{aligned} \gamma_1\gamma_2^x &\equiv (\alpha^{a_1}\beta^{a_2})\left(\alpha^{b_1}\beta^{b_2}\right)^x \ mod\ p \\ &\equiv \alpha^{a_1+xb_1}\beta^{a_2+xb_2} \ mod\ p \\ &= \alpha^{y_1}\beta^{y_2} \ mod\ p. \end{aligned}$$

□

Proposizione 13.30. Il sistema di firma di Pedersen-van Heyst è one-time.

Dimostrazione. Se $x \neq x'$ e

$$\begin{aligned} \text{sig}_K(x) &= (y_1, y_2) = ((a_1 + xb_1) \ mod\ q, (a_2 + xb_2) \ mod\ q) \\ \text{sig}_K(x') &= (y_1', y_2') = ((a_1 + x'b_1) \ mod\ q, (a_2 + x'b_2) \ mod\ q) \end{aligned}$$

allora

$$\begin{cases} y_1 = (a_1 + xb_1) \ mod\ q \\ y_2 = (a_2 + xb_2) \ mod\ q \\ y_1' = (a_1 + x'b_1) \ mod\ q \\ y_2' = (a_2 + x'b_2) \ mod\ q \end{cases}$$

Così

$$\begin{cases} a_1 = y_1 - x\,(y_1 - y_1')\,(x - x')^{-1} \mod q \\ b_1 = (y_1 - y_1')\,(x - x')^{-1} \mod q \\ a_2 = y_1' - x\,(y_1 - y_1')\,(x - x')^{-1} \mod q \\ b_2 = (y_2 - y_2')\,(x - x')^{-1} \mod q \end{cases}$$

e quindi $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ è nota, essendo $\gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \mod p$ e $\gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \mod p$.

□

Definizione 13.31. Siano $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ e $K' = (\gamma_1', \gamma_2', a_1', a_2', b_1', b_2')$ due chiavi nel sistema di firma di Pedersen-van Heyst, allora

$$K \sim K' \iff \gamma_1 = \gamma_1' \text{ e } \gamma_2 = \gamma_2'.$$

Quando ciò si verifca, diremo che K e K' sono equivalenti.

$\sim$ è una relazione di equivalenza sullo spazio delle chiavi $\mathcal{K}$.

Lemma 13.32. *Se $K \in \mathcal{K}$, allora $|[K]_\sim| = q^2$.*

Dimostrazione. Se $K, K' \in \mathcal{K}$ sono tali che $K \sim K'$, allora

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$
$$K' = (\gamma_1, \gamma_2, a_1', a_2', b_1', b_2').$$

Da (13.14) segue che

$$\begin{cases} \gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \mod p \equiv \alpha^{a_1'}\beta^{a_2'} \mod p \\ \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \mod p \equiv \alpha^{b_1'}\beta^{b_2'} \mod p \end{cases}$$

Siccome $G = \langle\alpha\rangle$ e $\beta \equiv \alpha^{a_0} \mod p$ allora

$$\begin{cases} \alpha^{a_1 + a_0 a_2} \equiv \alpha^{a_1' + a_0 a_2'} \mod p \\ \alpha^{b_1 + a_0 b_2} \equiv \alpha^{b_1' + a_0 b_2'} \mod p \end{cases}$$

che è equivalente a

$$\begin{cases} a_1 + a_0 a_2 \equiv a_1' + a_0 a_2' \mod q \\ b_1 + a_0 b_2 \equiv b_1' + a_0 b_2' \mod q \end{cases}$$

essendo $o(\alpha) = q$. Pertanto

$$K \sim K' \iff K' = (\gamma_1, \gamma_2, a_1 + a_0 a_2 - a_0 a_2', a_2', b_1 + a_0 b_2 - a_0 b_2', b_2'),$$

con $a_2', b_2' \in \mathbb{Z}_q$. Quindi, $|[K]_\sim| = q^2$.

□

Lemma 13.33. *Se $K \sim K'$, allora*

$$\mathrm{ver}_K(x,y) = vero \iff \mathrm{ver}_{K'}(x,y) = vero.$$

Dimostrazione. Se $K, K' \in \mathcal{K}$ sono tali che $K \sim K'$, allora

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$
$$K' = (\gamma_1, \gamma_2, a_1', a_2', b_1', b_2')$$

e quindi

$$\begin{cases} \gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ mod\ p \equiv \alpha^{a_1'}\beta^{a_2'} \ mod\ p \\ \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ mod\ p \equiv \alpha^{b_1'}\beta^{b_2'} \ mod\ p \end{cases}$$

Pertanto,

$$\mathrm{ver}_K(x,y) = vero \Leftrightarrow \gamma_1\gamma_2^x \equiv \alpha^{y_1}\beta^{y_2} \ mod\ p \Leftrightarrow \mathrm{ver}_{K'}(x,y) = vero.$$

□

Lemma 13.34. *Sia K una chiave e $y = \mathrm{sig}_K(x)$, allora esistono esattamente q chiavi K' tali che $K' \sim K$ e $y = \mathrm{sig}_{K'}(x)$.*

Dimostrazione. Determinare il numero delle chiavi $K' \sim K$ e $y = \mathrm{sig}_{K'}(x)$ equivale a determinare in corrispondenza della chiave pubblica (γ_1, γ_2) il numero delle quadruple (a_1, a_2, b_1, b_2) che sono soluzioni del sistema

$$\begin{cases} \gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ mod\ p \\ \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ mod\ p \\ y_1 \equiv a_1 + xb_1 \ mod\ q \\ y_2 \equiv a_2 + xb_2 \ mod\ q. \end{cases} \tag{13.15}$$

Siccome, $G = \langle\alpha\rangle$, esistono (unici) $c_1, c_2, a_0 \in \mathbb{Z}_q$ tali che

$$\begin{cases} \gamma_1 \equiv \alpha^{c_1} \ mod\ p \\ \gamma_2 \equiv \alpha^{c_2} \ mod\ p \\ \beta \equiv \alpha^{a_0} \ mod\ p, \end{cases}$$

quindi (13.15) è equivalente a

$$\begin{cases} c_1 \equiv a_1 + a_0a_2 \ mod\ q \\ c_2 \equiv b_1 + a_0b_2 \ mod\ q \\ y_1 \equiv a_1 + xb_1 \ mod\ q \\ y_2 \equiv a_2 + xb_2 \ mod\ q. \end{cases} \tag{13.16}$$

che in notazione matriciale è

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix} \tag{13.17}$$

Siccome K è una chiave che realizza $y = \text{sig}_K(x)$, allora il sistema (13.17) è compatibile. Se R_i denota l'i-sima riga della matrice del sistema (13.17), si ha che

$$R_1 + xR_2 - R_3 - a_0R_4 = O$$

e siccome R_1, R_2 e R_3 sono linearmente indipendenti, allora il rango della matrice del sistema (13.17) è 3. Il **Teorema di Rouché-Capelli** implica che (13.17) a coefficienti nel campo $\mathbb{Z}_q$ (q è primo) ammette $q^{4-3} = q$ soluzioni. Cioè q è il numero delle chiavi K' tali che $K' \sim K$ e $y = \text{sig}_{K'}(x)$.

□

Lemma 13.35. *Sia K una chiave e $y = \text{sig}_K(x)$. Se $\text{ver}_K(x', y') = vero$ con $x' \neq x$, allora esiste al più un'altra chiave K' tale che $K' \sim K$ e $y = \text{sig}_{K'}(x)$ e $y' = \text{sig}_{K'}(x')$.*

Dimostrazione. Determinare il numero delle chiavi $K' \sim K$ e $y = \text{sig}_{K'}(x)$ e $y' = \text{sig}_{K'}(x')$, sapendo che $\text{ver}_K(x', y') = vero$ con $x' \neq x$, equivale a determinare in corrispondenza della chiave pubblica (γ_1, γ_2) il numero delle quadruple (a_1, a_2, b_1, b_2) che sono soluzioni del sistema

$$\begin{cases} \gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \ mod\ p \\ \gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \ mod\ p \\ y_1 \equiv a_1 + xb_1 \ mod\ q \\ y_2 \equiv a_2 + xb_2 \ mod\ q \\ \gamma_1\gamma_2^{x'} = \alpha^{y_1'}\beta^{y_2'} \ mod\ p \\ y_1' \equiv a_1 + x'b_1 \ mod\ q \\ y_2' \equiv a_2 + x'b_2 \ mod\ q. \end{cases} \tag{13.18}$$

Poiché $y' = \text{sig}_{K'}(x')$, allora $\text{ver}_{K'}(x', y') = vero$. Dal **Lemma 13.33** segue che $\text{ver}_K(x', y') = vero$, quindi la quinta equazione del sistema (13.18) è sovrabbondante.
Siccome, $G = \langle\alpha\rangle$, esistono (unici) $c_1, c_2, a_0 \in \mathbb{Z}_q$ tali che

$$\begin{cases} \gamma_1 \equiv \alpha^{c_1} \ mod\ p \\ \gamma_2 \equiv \alpha^{c_2} \ mod\ p \\ \beta \equiv \alpha^{a_0} \ mod\ p, \end{cases}$$

quindi (13.18) è equivalente a

$$\begin{cases} c_1 \equiv a_1 + a_0a_2 \ mod\ q \\ c_2 \equiv b_1 + a_0b_2 \ mod\ q \\ y_1 \equiv a_1 + xb_1 \ mod\ q \\ y_2 \equiv a_2 + xb_2 \ mod\ q. \\ y_1' \equiv a_1 + x'b_1 \ mod\ q \\ y_2' \equiv a_2 + x'b_2 \ mod\ q. \end{cases} \tag{13.19}$$

che in notazione matriciale è

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \\ 1 & 0 & x' & 0 \\ 0 & 1 & 0 & x' \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \\ y_1' \\ y_2' \end{pmatrix} \tag{13.20}$$

Siccome $x \neq x'$, allora il rango della matrice del sistema è 4, per il **Teorema di Rouché-Capelli**, il sistema (13.20) è compatibile, ed in tal caso ammette un'unica soluzione se, e solo se, la matrice completa ha anche rango 4.

□

Teorema 13.36. *Se* $\text{sig}_K(x) = y$ *e* $x' \neq x$ *sono noti, allora la probabilità che un avversario calcoli* $\text{sig}_K(x')$ *è al più* $1/q$.

Dimostrazione. Se $\text{sig}_K(x) = y$ e $x' \neq x$ sono noti, per il **Lemma 13.34** ci sono esattamente q chiavi K' tali che $K' \sim K$ e $y = \text{sig}_{K'}(x)$ e, per il **Lemma 13.33**, al più una di esse è tale che $y' = \text{sig}_{K'}(x')$. Quindi, la probabilità che un avversario calcoli $y' = \text{sig}_{K'}(x')$ è al più $1/q$.

□

Il risultato contenuto nel **Teorema 13.36** non dipende dalle risorse computazionali dell'avversario, siccome quest'ultimo non riesce a dire quale delle q possibili chiavi è stata usata dall'utente firmatario. Pertanto, la sicurezza è incondizionata.

Analizziamo il concetto fail-stop. Supponiamo che ad un utente A gli venga fornita una coppia (x', y'') tale che

$$\begin{cases} \text{sig}_K(x') \neq y'' \\ \text{ver}_K(x', y'') = vero \end{cases}$$

cioè che

$$\begin{cases} \text{sig}_K(x') \neq y'' = (y_1'', y_2'') \\ \gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1''} \beta^{y_2''} \ mod\ p \end{cases} \tag{13.21}$$

Allora l'utente A calcola la propria firma su x', quindi

$$\begin{cases} \text{sig}_K(x') = y' = (y_1', y_2') \\ \gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1'} \beta^{y_2'} \ mod\ p. \end{cases} \tag{13.22}$$

Allora da (13.21) e (13.22), segue che

$$\alpha^{y_1''} \beta^{y_2''} \equiv \alpha^{y_1'} \beta^{y_2'} \ mod\ p.$$

Siccome $\beta \equiv \alpha^{a_0} \ mod\ p$, si ha

$$\alpha^{y_1'' + a_0 y_2''} \equiv \alpha^{y_1' + a_0 y_2'} \ mod\ p,$$

o equivalentemente

$$y_1'' + a_0 y_2'' \equiv y_1' + a_0 y_2' \ mod\ q \Longleftrightarrow y_1'' - y_1' \equiv a_0 (y_2' - y_2'') \ mod\ q \tag{13.23}$$

Siccome $y'' \neq y'$, da (13.23) segue che $y_1'' \neq y_1'$ e $y_2' \neq y_2''$. Pertanto, $y_2' - y_2''$ è invertibile in $\mathbb{Z}_q^*$ e quindi

$$a_0 = \log_\alpha \beta = (y_1'' - y_1')(y_2' - y_2'')^{-1} \ mod\ q \tag{13.24}$$

Siccome a_0 è noto solo all'ente di fiducia, poichè stiamo assumendo implicitamente che A non sia in grado di calcolare il logaritmo discreto $\log_\alpha \beta$. Quindi, il fatto che A sia in grado di determinare a_0 è una dimostrazione dell'avvenuta falsificazione.

Esempio 13.37. Sia $p = 3467 = 2 \times 1733 + 1$, e sia $\alpha = 4$ un elemento di ordine 1733 in $\mathbb{Z}_{3467}^*$. Sia $a_0 = 1567$, allora

$$\beta = 4^{1567} \ mod\ 3467 = 514.$$

Quindi, i valori scelti dall'ente di fiducia sono

$$(p, q, \alpha, a_0, \beta) = (3467, 1733, 4, 1567, 514),$$

i valori $(p, q, \alpha, \beta) = (3467, 1733, 4, 514)$ sono pubblici, invece $a_0 = 1567$ è segreto. Un utente A crea la sua firma usando $(a_1, a_2, b_1, b_2) = (888, 1024, 786, 999)$, quindi

$$\begin{aligned} \gamma_1 &= 4^{888} 514^{1024} \ mod\ 3467 = 3405 \\ \gamma_2 &= 4^{786} 514^{999} \ mod\ 3467 = 2281, \end{aligned}$$

$(\gamma_1, \gamma_2) = (3405, 2281)$ è pubblica, invece $(a_1, a_2, b_1, b_2) = (888, 1024, 786, 999)$ è privata.

Supponiamo che all'utente A sia inviato il messaggio $x = 3833$ munito di firma **falsificata** $(822, 55)$. La firma è valida perché la condizione di verifica è soddisfatta:

$$\begin{aligned} \gamma_1 \gamma_2^x &= 3405 \times 2281^{3833} \ mod\ 3467 = 2282 \\ \alpha^{y_1} \beta^{y_2} &= 4^{822} \times 514^{55} \ mod\ 3467 = 2282. \end{aligned}$$

Tuttavia, questa non è la firma che l'utente A avrebbe costruito.
Infatti, A produce la sua firma come segue:

$$\begin{aligned} a_1 + x b_1 \ mod\ q &= (888 + 3833 \times 786) \ mod\ 1733 = 1504 \\ a_2 + x b_2 \ mod\ q &= (1024 + 3833 \times 999) \ mod\ 1733 = 1291. \end{aligned}$$

Quindi la firma che A allega al messaggio $x = 3833$ è la coppia $(1504, 1291)$. Con queste informazioni, A calcola il logaritmo discreto con (13.24)

$$a_0 = \log_\alpha \beta = (8232 - 1504)(1291 - 55)^{-1} \ mod\ 1733 = 1567$$

che dovrebbe essere segreto. Questa è la prova dell'avvenuta falsificazione. Infatti, a_0 è noto solo all'ente di fiducia, siccome $a_0 = \log_\alpha \beta$ è computazionalemnte intrattabile.

□

Bibliografia

[1] O. Goldreich, *Foundations of cryptography*, Cambridge University Press, Cambridge, 2001.

[2] J. Katz, Y. Lindell, Yehuda, *Introduction to modern cryptography*, Second edition, Chapman & Hall/CRC Cryptography and Network Security, 2015.

[3] N. Koblitz, *A course in number theory and cryptography*, Second edition, Graduate Texts in Mathematics, **114**, Springer-Verlag, New York, 1994.

[4] D. R. Stinson, *Cryptography. Theory and practice*, Third edition, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2006.

[5] L.C. Washington, *Elliptic curves. Number theory and cryptography*, Second edition, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2008.