

# LA SFIFA FINALE



## SCOPO DELL'ESPERIMENTO

L'esperimento consiste nella decifrazione di un messaggio segreto nascosto dentro un'immagine mediante l'ausilio di un robot telecomandato e tecniche di crittoanalisi.

## MATERIALI E STRUMENTI

- Il reperto: cinque immagini (formato bitmap) con le foto dei soliti sospetti (Paul, Max, Yuri, Axel e Caietanus) contenenti un messaggio col nome del colpevole, nascosto mediante la tecnica della steganografia.
- Prima parte:
  - Robot LEGO Mindstorm EV3 + smartphone con app (*Phone eye*) per trasmettere in streaming la videocamera su un'indirizzo I
  - Percorso a ostacoli realizzato mediante scatoloni
  - Notebook/Tablet con software per LEGO Mindstorm EV3
  - Altro smartphone con app per leggere i codici QR (*QR Droid*, *QR Reader*, ...)



Fig. 1 Robot LEGO Mindstorm EV3

- Seconda parte
  - PC con software *Mathematica* per decrittare la password contenuta nel codice QR
- Terza parte
  - PC con software *S-Tools* per decifrare il messaggio segreto nascosto nelle immagini

## PRIMA PARTE - IL ROBOT TELECOMANDATO → CODICE QR → PASSWORD CRITTATA

### **Breve richiamo teorico- Il codice QR**

Un codice QR è una figura a barre bidimensionale composta da moduli neri disposti all'interno di uno schema di forma quadrata. Viene impiegato per memorizzare informazioni generalmente destinate a essere lette tramite un dispositivo mobile (smartphone, tablet,...).

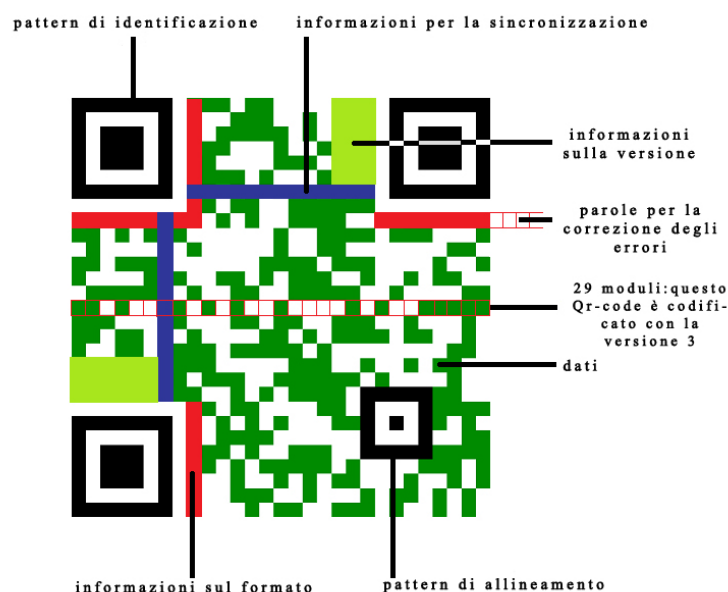
Il nome QR è l'abbreviazione dell'inglese *Quick Response* (risposta rapida), in virtù del fatto che il codice fu sviluppato per permettere una rapida decodifica del suo contenuto.

Il codice QR fu sviluppato nel 1994 dalla compagnia giapponese Denso Wave, allo scopo di tracciare i pezzi di automobili nelle fabbriche di Toyota; vista la sua capacità di contenere più dati di un codice a barre, venne in seguito utilizzato per la gestione delle scorte da diverse industrie.

Nel 1999 Denso Wave ha rilasciato i codici QR sotto licenza libera, favorendone così la diffusione in Giappone, e poi in tutto il mondo. In parallelo al diffondersi del web-mobile nella vita quotidiana i codici QR iniziano ad apparire su giornali, riviste, cartelloni pubblicitari, etichette dei prodotti, etc...

consentendo agli utenti di ricevere maggiori informazioni sul prodotto pubblicizzato semplicemente inquadrando il codice con la telecamera del proprio smartphone. È proprio questo il settore nel quale ancora oggi i codici Qr stanno riscontrando maggior successo: poiché sono in grado di codificare un abbondante numero di caratteri alfanumerici sono per lo più utilizzati per veicolare un indirizzo web, un URL. L'utente, dopo aver aperto l'applicazione che legge il codice, lo fotografa con la telecamera del proprio smartphone, che lo decodifica e apre automaticamente il browser all'indirizzo codificato. In questo modo si evita il fastidioso compito di inserire dati da tastiera o da touch-screen, e si garantisce un link rapido a informazioni aggiuntive e multimediali, allargando gli orizzonti del marketing pubblicitario.

Esistono diverse versioni di codici QR, ognuna con capacità e dimensioni diverse; al massimo essi possono memorizzare fino a un massimo di 4.296 caratteri alfanumerici o 7.089 caratteri numerici. L'informazione è distribuita sia in orizzontale che in verticale, e ci sono dei pattern specifici di identificazione della posizione e dell'allineamento che consentono la decodifica anche se l'immagine è ruotata o distorta.



**Figura 2** Struttura di un codice QR

Nei codici QR è utilizzato il metodo Reed-Solomon per la rilevazione e la correzione degli errori: nel caso in cui il QR fosse in parte danneggiato, per esempio da macchie o graffi sul supporto cartaceo, la procedura Reed-Solomon permette di ricostruire i dati persi, ripristinando, durante la decodifica, fino al 30% delle informazioni codificate.

Sfruttando la capacità di rilevazione e correzione d'errore Reed-Solomon dei codici QR, si possono modificare i codici entro il limite della leggibilità, incorporando immagini, loghi, caratteri e foto, senza perdere alcuna informazione utile alla lettura del codice



**Fig. 3** Esempi di codici QR

### Esecuzione dell'esperimento e analisi dati

- Costruire un robot LEGO Mindstorm EV3 in modo che possa reggere uno smartphone, con cui riprendere la scena che si trova davanti.
- Attivare la videocamera dello smartphone sul robot e avviare l'app Phone eye, aprire sull'altro smartphone la pagina internet indicata (digitando sul browser l'indirizzo internet "http://..." che compare sul display dello smartphone sul robot) e verificare che sul display venga trasmesso in streaming il video ripreso.
- Mediante il software LEGO del notebook/tablet programmare a distanza il robot in modo che effettui il percorso a ostacoli assegnato, e raggiunga la posizione in cui c'è un documento con un codice QR.
- Quando sul display è visibile l'immagine del codice QR, fare uno screenshot, decifrare il suo contenuto mediante un'app specifica (*QR Droid, QR Reader, ...*) e annotarlo → Questa è la password che dovrà essere decrittata mediante il software Mathematica..

### SECONDA PARTE - CRITTOANALISI CON MATHEMATICA → PASSWORD PER STEGANOGRAFIA

#### Breve richiamo teorico – Il Codice di Vigenère

Il *cifrario di Vigenère* è il classico esempio di metodo di sostituzione polialfabetica; fu inventato da *Blaise de Vigenère* nel 1586 e ritenuto per secoli inattaccabile. Questo algoritmo di crittazione ebbe una fortuna immediata e fu molto usato nell'ambito militare, anche dopo che fu pubblicato un procedimento per la sua decifrazione (Fredrich Kasiski, 1863). Esso si può considerare una generalizzazione del *cifrario di Cesare*; invece di spostare sempre dello stesso numero di posti la lettera da cifrare, questa viene spostata di un numero di posti variabile ma ripetuto, determinato in base ad una parola chiave, da concordarsi tra mittente e destinatario, e da scrivere ripetutamente sotto il messaggio, carattere per carattere; la chiave era detta anche *verme*, per il motivo che, essendo in genere molto più corta del messaggio, deve essere ripetuta molte volte sotto questo, come nel seguente esempio:



Fig. 4 Blaise de Vigenère

<i>Testo chiaro:</i>	R	A	P	P	O	R	T	O	I	M	M	E	D	I	A	T	O
<i>Parola chiave:</i>	V	E	R	M	E	V	E	R	M	E	V	E	R	M	E	V	E
<i>Testo cifrato:</i>	M	E	G	B	S	M	X	F	U	Q	H	I	U	U	E	O	S

Il testo cifrato si ottiene spostando la lettera chiara di un numero fisso di caratteri, pari al numero ordinale della lettera corrispondente della parola chiave. Di fatto si esegue una somma aritmetica tra l'ordinale del chiaro (A = 0, B = 1, C = 2, ... M = 12, ... R = 17, ... V = 21, ... Y = 24, Z = 25) e quello della chiave; se si supera l'ultima lettera, la Z, si ricomincia dalla A, secondo la logica dell'aritmetica modulare (di modulo 26, perché 26 è il numero totale delle lettere dell'alfabeto romano):

$$R + V \rightarrow 17 + 21 = 38 \equiv 12 \pmod{26},$$

cioè 38 è congruente a 12 modulo 26, perché  $\frac{38}{26} = 1 + \frac{12}{26}$ , cioè la divisione intera tra 38 e 26 ha come quoziente 1 e come resto 12, e il numero 12 corrisponde alla lettera M, quindi  $R + V \rightarrow M$ .

Il vantaggio rispetto ai cifrari monoalfabetici (come il cifrario di Cesare o quelli per sostituzione delle lettere con simboli/altre lettere) è evidente: il testo è cifrato con n alfabeti cifranti. In questo modo, la stessa lettera viene crittata (se ripetuta consecutivamente) n volte; ciò rende quindi più complessa la crittoanalisi del testo cifrato.

Per semplificare la cifratura, Vigenère propose l'uso della seguente tavola quadrata (vedi fig. 5), composta da alfabeti ordinati e spostati. Se si vuole cifrare, con la chiave dell'esempio precedente, la lettera "R" della parola RAPPORTO basterà trovare la lettera "R" nella prima riga, la lettera "V" nella prima colonna, e individuare, tramite l'incrocio, la lettera cifrata da usare, cioè la M.

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

Fig. 5 Tavola quadrata di Vigenère

Chi riceve il messaggio per decifrarlo deve semplicemente usare il metodo inverso (sottrarre invece che sommare, sempre mediante le regole dell'aritmetica modulare); riferendosi all'esempio di sopra si avrà:

$$M - V \rightarrow 12 - 21 = -9 \equiv 17 \pmod{26},$$

cioè -9 è congruente a 17 modulo 26, perché  $\frac{-9}{26} = -1 + \frac{17}{26}$ , cioè la divisione intera tra -9 e 26 ha come quoziente -1 e come resto 17, e il numero 17 corrisponde alla lettera R, quindi  $M - V \rightarrow R$

Testo cifrato:	M	E	G	B	S	M	X	F	U	Q	H	I	U	U	E	O	S
Parola chiave:	V	E	R	M	E	V	E	R	M	E	V	E	R	M	E	V	E
Testo chiaro:	R	A	P	P	O	R	T	O	I	M	M	E	D	I	A	T	O

La debolezza del codice di Vigenère sta nell'essere, di fatto, un insieme di n cifrari di Cesare, dove n è la lunghezza della chiave; se il crittoanalista riesce a determinare la lunghezza della chiave (nel nostro caso, n) la decrittazione diventa semplice.

Per ovviare a questo problema, si è provato a scegliere chiavi molto lunghe, (e quindi, non ripetute o solo in minima parte). Così facendo, gli alfabeti cifranti sono molti di più e le probabilità di ripetizioni sono inferiori.

Un cifrario di Vigenere molto più sicuro dell'originale è quello che usa una *tavola disordinata* nel senso che invece di usare alfabeti ordinati, usa alfabeti con le lettere disposte in ordine casuale; cifratura e decifrazione avvengono come per il cifrario di Vigenère classico, ma in questo caso la crittoanalisi statistica diventa molto più difficile. In questo caso la segretezza si basa non solo sulla

chiave (verme) ma anche sulla tavola, che va custodita con la massima attenzione; se questa cade nelle mani del nemico, la sicurezza di questo cifrario è compromessa.

Un cifrario di questo tipo fu usato durante la seconda guerra mondiale dagli agenti dello *Special Operations Executive (SOE)* uno speciale reparto dei servizi segreti britannici, creato da Winston Churchill nel 1940 con il compito di compiere operazioni di sabotaggio, sovversione e supporto ai gruppi della resistenza nell'Europa occupata dai nazisti; "*E ora incendiate l'Europa*" fu l'incitamento iniziale di Churchill all'SOE. La tavola era scritta su fazzoletti di seta, mentre le frequenze radio ed eventuali chiavi e disposizioni transitorie erano scritte su carta di riso. L'agente che fosse stato catturato dal nemico, doveva immediatamente incendiare il fazzoletto (la seta brucia molto rapidamente), oppure ... fare un sol boccone della carta di riso.

### Esecuzione dell'esperimento e analisi dati

Aprire il software Mathematica e scrivere un algoritmo per decrittare la password contenuta nel codice QR, cifrata mediante una variazione del codice di Vigenere. Nel nostro caso abbiamo di fronte un algoritmo di crittazione praticamente inattaccabile, in quanto la chiave di crittazione contiene lo stesso numero di caratteri della stringa crittata. Utilizzeremo allora il sistema UNICODE per sostituire i caratteri di una stringa da crittare con un numero, da sommare, sottrarre, moltiplicare, etc. ai numeri proveniente, nello stesso codice, dalla chiave. I computer non sanno far altro che trattare numeri, in forma binaria. Per immagazzinare in memoria lettere o altri segni è necessario che a ogni carattere venga assegnato un numero. Esistono centinaia di sistemi di codifica, preesistenti a UNICODE, e ognuno di questi abbina i numeri ai caratteri in modo differente. Nessuna di queste codifiche comprende un numero di caratteri sufficiente per tutte le circostanze. Per le sole lingue dell'Unione Europea, ad esempio, è necessario utilizzare parecchi sistemi di codifica distinti. Anche considerando una sola lingua, come l'italiano, non esiste una codifica unica che comprenda tutte le lettere e tutti i segni di punteggiatura e simboli tecnici di uso comune. Questi diversi sistemi di codifica, inoltre, sono in contraddizione l'uno con l'altro. Succede che due codifiche utilizzino lo stesso numero per due caratteri diversi o che, viceversa, adottino numeri diversi per lo stesso carattere. Qualsiasi elaboratore, e a maggior ragione un server di rete, ha bisogno di utilizzare codifiche diverse. Il problema è che, quando i dati passano da una codifica a un'altra, o da una piattaforma a un'altra, si corre il serio rischio di perdere informazioni. Per questo motivo, occorre scegliere inizialmente un sistema di codifica, l'UNICODE per l'appunto.

Latino base																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	[NOC]	[BOI]	[SFR]	[ETR]	[BOT]	[ENG]	[ADR]	[BEL]	[GER]	[HY]	[LP]	[VR]	[FR]	[GR]	[SS]	[SI]
001	[BLE]	[CZ]	[DAN]	[DEU]	[DUA]	[NAR]	[SNA]	[FIS]	[CAN]	[SM]	[SAB]	[ESE]	[RS]	[SS]	[RS]	[US]
002	[SP]	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{		}	~	[DEC]
Latino esteso																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
008	[PAG]	[ROB]	[BPA]	[RBA]	[IND]	[NEC]	[SSA]	[ESA]	[FRS]	[FR]	[FRS]	[PDS]	[PDU]	[RI]	[SSS]	[SSS]
009	[DSS]	[PDI]	[PDI]	[SFS]	[DGN]	[MW]	[SAR]	[EFA]	[SOS]	[SSS]	[SST]	[DS]	[SF]	[SSS]	[RAT]	[ABC]
00A	[NBSP]	ı	ç	£	¤	¥	ı	§	¨	©	ª	«	¬	[SFA]	®	—
00B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
00C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
00D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
00E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
00F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Fig. 6 Tabella dei caratteri UNICODE intervallo (0000-FFFF)

### *Caratteristiche generali del linguaggio Mathematica*

Nel software *Mathematica* ogni input che l'utente fornisce al computer viene indicato da un nome a scelta dell'utente. Gli input possono essere variabili numeriche o stringhe di testo. Ad esempio:

```
x=15;  
(*dove x è una variabile numerica*)  
cyphermess="ubgtvgowwevmdhc dynvwind eezxxcf"  
(*dove cyphermess è una stringa*)
```

È importante terminare ogni istruzione con il ";" in modo che il programma non la ripeta sullo schermo in fase di esecuzione. Per visualizzare il risultato, naturalmente, basterà togliere il ";" alla fine dell'istruzione. Gli *input* possono essere anche costituiti da *vettori* o *matrici* di *dati numerici*. Ad esempio, mediante il comando *Table* e un solo indice (*k*, che varia nell'intervallo *1,10*, con passo *1*) si può scrivere la seguente istruzione:

```
input=Table[k^2,{k,1,10,1}];
```

che genera un vettore di dimensione *10*, contenente i quadrati dei primi *10* numeri naturali:

```
{1,4,9,16,25,36,49,64,81,100}
```

Per determinare, all'interno di un vettore, uno specifico elemento si utilizzano le doppie parentesi quadre. Ad esempio:

```
input[[3]]=9;
```

Il comando **Characters** trasforma la frase da decriptare in una **Table** contenente i singoli caratteri che la compongono. Ad esempio:

```
cyphermess="ubgtvgowwevmdhc dynvwind eezxxcf";  
cyphersplit=Characters[cyphermess];  
cyphersplit={u, b, g, t, v, g, o, w, w, e, v, m, d, h, c, , d, y, n, v, w, i, n, d, , e, e,  
z, x, x, c, f}
```

che costituisce un array di caratteri.

### *Costruzione dell'algoritmo*

La chiave di decrittazione si ottiene unendo i codici alfanumerici ottenuti dai quattro esperimenti svolti, rispettando il seguente ordine:

- 1) analisi delle impronte digitali: nome della persona a cui appartiene l'impronta → per es. "Dave"
- 2) pendolo balistico: codice dell'arma → per es. "H57"
- 3) riconoscimento audio: codice segreto di quattro caratteri → per esempio "XZGT"
- 4) analisi documenti contraffatti: codice segreto di quattro caratteri → per esempio "ABKW"

In questo caso bisogna far diventare delle stringhe i quattro codici ottenuti, scrivendo:

```
esp1 = "Dave"  
esp2 = "H57"  
esp3 = "XZGT"  
esp4 = "ABKW"
```

Poi, per ottenere la chiave, bisogna combinare le quattro stringhe in un unico array di 15 caratteri attraverso le istruzioni **Characters** e **StringJoin**:

```
(*Ricostruiamo la stringa e formiamo così la key*)
key = Characters[StringJoin[esp1, esp2, esp3, esp4]]];
```

Ora bisogna fare la stessa operazione con la password crittata ottenuta dal codice QR, ad esempio:

```
(*Frase ottenuta dal codice QR*)
codiceqr = Characters["SEMPRECAROMIFU!"]];
```

A questo punto avrete ottenuto due array contenenti lo stesso numero di caratteri nella forma seguente:

{ D, a, v, e, H, 5, 7, X, Z, G, T, A, B, K, W}      {S, E, M, R, E, C, A, R, O, M, I, F, U, !}

La prima operazione è quella di convertire i due array nelle loro componenti UNICODE. Questa operazione deve avvenire carattere per carattere, mediante l'istruzione **ToCharacterCode**, che produce in uscita il numero UNICODE corrispondente ad un carattere. Ad esempio, il numero UNICODE corrispondente alla A (la A è una stringa e va messa tra le virgolette), è 65:

```
ToCharacterCode["A"]
{ 65 }
```

Per tradurre le due stringhe di 15 caratteri in UNICODE utilizziamo una serie di istruzioni annidate:

L'istruzione Table permette al matematico di ricostruire un array numerico corrispondente al numero dei caratteri della stringa, 15 in questo caso

Questa istruzione traduce il singolo carattere nel suo formato UNICODE

Identifica l'elemento k-esimo nell'array di caratteri codiceqr

```
codecodiceqr = Flatten[Table[ToCharacterCode[codiceqr[[k]]], {k, Length[codiceqr]}]]
{83, 69, 77, 80, 82, 69, 67, 65, 82, 79, 77, 73, 70, 85, 33}
```

Questa è la stringa codiceqr tradotta in formato UNICODE

Ricordatevi che l'elemento di un array deve essere identificato da un numero compreso tra 1 e la lunghezza dell'array (Length[codiceqr] in questo caso), posto tra due coppie di parentesi quadre [...]

Il successivo passo da fare verso la decodifica della stringa è molto semplice: dobbiamo sommare tra loro i valori UNICODE delle due stringhe, presi in modo opposto, cioè il primo della chiave con l'ultimo elemento della table ottenuta dal codice QR, il secondo della chiave con il penultimo della table QR, etc. e poi moltiplicare tra loro i risultati ottenuti (si avrà quindi una moltiplicazione tra 15 numeri interi positivi).

Questa operazione può essere fatta semplicemente attraverso un **ciclo Do**:

```

(*Creiamo l'agoritomo di crittazione: i codici Unicode della key e della stringa QR,
vengono sommati tra loro in modo invertito
(il primo della key con l'ultimo della stringa QR,
il secondo della key con il penultimo della stringa QR, etc.). I risultati sono
poi moltiplicati tra loro*)
prod = 1;
Do[
  prod = prod * (codekey[[k]] + codecodiceqr[[Length[codecodiceqr] - k + 1]])
, {k, Length[key]}]

```

Definiamo una variabile prodotto, che ci servirà all'interno del ciclo per definire il prodotto delle somme dei termini ad ogni step del ciclo

Definiamo il ciclo, il numero dei passi da fare è definito dalla variabile k che va da 1 alla lunghezza delle due stringhe da sommare (Length[key] in questo caso)

Ricordatevi di invertire l'ordine dei numeri ottenuti dalla stringa QR: dovete prenderli dall'ultimo al primo!

Questa riga definisce le operazioni da fare con i due array dei numeri UNICODE

Se le operazioni sono state eseguite in modo corretto, alla fine del ciclo la variabile **prod** sarà un numero davvero molto grande, tipo:

459116907246947113395626723328000

L'ultima operazione da fare sarà quella di esprimerlo in una base iper decimale, cioè contenente un numero di simboli maggiori di 10. La più famosa base iper decimale è quella esadecimale, che utilizza 16 simboli, i 10 numeri + 6 lettere, dalla A alla F

	BINARY	HEXADECIMAL	
$2^0$	0	0	$16^0$
	1	1	
$2^1$	10	2	
	11	3	
$2^2$	100	4	
	101	5	
	110	6	
	111	7	
$2^3$	1000	8	
	1001	9	
	1010	A	
	1011	B	
	1100	C	
	1101	D	
	1110	E	
	1111	F	
$2^4$	10000	10	$16^1$
	10001	11	
	10010	12	
	10011	13	
	10100	14	
	10101	15	
	10110	16	
	10111	17	
	11000	18	
	11001	19	
	11010	1A	
	11011	1B	
	11100	1C	

La nostra base sarà composta da 36 simboli, 10 numerici e 26 lettere, come si vede dal seguente schema:

0 <sub>36</sub>	1 <sub>36</sub>	2 <sub>36</sub>	3 <sub>36</sub>	4 <sub>36</sub>	5 <sub>36</sub>
6 <sub>36</sub>	7 <sub>36</sub>	8 <sub>36</sub>	9 <sub>36</sub>	a <sub>36</sub>	b <sub>36</sub>
c <sub>36</sub>	d <sub>36</sub>	e <sub>36</sub>	f <sub>36</sub>	g <sub>36</sub>	h <sub>36</sub>
i <sub>36</sub>	j <sub>36</sub>	k <sub>36</sub>	l <sub>36</sub>	m <sub>36</sub>	n <sub>36</sub>
o <sub>36</sub>	p <sub>36</sub>	q <sub>36</sub>	r <sub>36</sub>	s <sub>36</sub>	t <sub>36</sub>
u <sub>36</sub>	v <sub>36</sub>	w <sub>36</sub>	x <sub>36</sub>	y <sub>36</sub>	z <sub>36</sub>

L'istruzione da usare è **BaseForm**, con la seguente sintassi:

(\*Esprimiamo il risultato in una base a 36 cifre\*)  
**BaseForm**[prod, 36]

```
Out[22]//BaseForm=
ycg4amje703zzptzb400036
```

In questo caso, la serie di caratteri da utilizzare come password per scoprire il messaggio nascosto nelle immagini mediante steganografia sarebbe → **ycg4amje703zzptzb4000**

### TERZA PARTE – ANALISI STEGANOGRAFICA → L'ASSASSINO E'....

#### Breve richiamo teorico

Il termine *steganografia* deriva dalle parole greche *στεγανός* (coperto) e *γραφία* (scrittura), ed indica l'insieme di tecniche che consentono di nascondere messaggi, che devono essere intelligibili al solo destinatario, inserendoli all'interno di un contesto del tutto estraneo, che funge da contenitore, in grado di nascondere la stessa esistenza della comunicazione, agli occhi di un eventuale osservatore. La steganografia è una forma di sicurezza tramite segretezza.

Tracce di questa tecnica si hanno già nell'antica Grecia: Erodoto narra l'episodio di Demarato di Sparta che per avvisare i compatrioti di una possibile invasione persiana scrive su di una tavoletta un messaggio da nascondere, poi copre la tavoletta di cera e sulla cera scrive un messaggio innocuo; dato che nell'antichità le tavolette di cera erano normalmente usate per scrivere testi provvisori, non destò sospetti.

Nella moderna steganografia digitale i messaggi vengono nascosti all'interno di immagini, file audio e video mediante diverse procedure; la più diffusa è la tecnica sostitutiva LSB (Least Significant Bit), che si basa sulla teoria secondo la quale l'aspetto di un'immagine digitale ad alta definizione non cambia se i colori vengono modificati in modo impercettibile.

Ogni pixel di un'immagine bitmap è rappresentato da una terna di numeri (le coordinate **RGB**) che esprimono il livello dell'intensità dei colori fondamentali (**Red, Green e Blu**); cambiando i bit meno significativi di queste terne, il singolo colore non risulterà variato in modo significativo e il contenuto dell'immagine sarà preservato nonostante questa manipolazione (vedi figure seguenti).

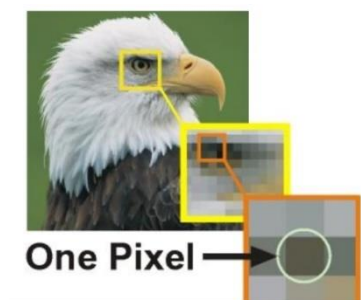


Fig. 7 I pixel di un'immagine digitale

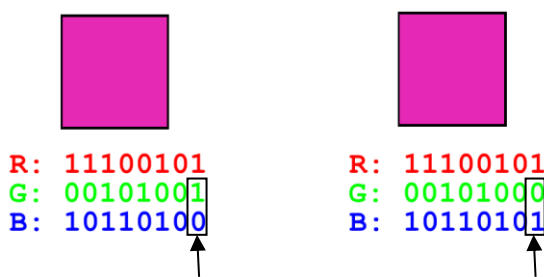


Fig. 8 Tecnica LSB nelle coordinate RGB di un pixel

L'algoritmo steganografico riceve in input un'immagine di copertura (C), una chiave (K) ed un dato da nascondere (D). Estrae da C i bit meno significativi, misurandone la dimensione e verificando se questa è sufficiente per ospitare D; tramite la chiave K, il dato D viene sparpagliato tra i bit meno significativi, sovrascrivendo i valori originali. Viene così generato S, l'immagine steganografica (vedi figure seguenti).



Fig. 9 Immagine di copertura (C)



Fig. 10 Immagine steganografica

Chi analizzerà S avrà davanti a sé un'immagine plausibile, e pur conoscendo l'algoritmo avrà bisogno della chiave K per verificare se c'è un messaggio nascosto.

L'esempio portato con le immagini può essere replicato su altri dati multimediali (file audio e video), perché tutti condividono la caratteristica di tollerare distorsioni plausibili, lasciando fruibile il contenuto originale.

#### ESECUZIONE DELL'ESPERIMENTO E ANALISI DATI

- Aprire il software S Tools, trascinare il file bmp contenente l'immagine del sospetto nella finestra principale di S-Tools.
- Selezionare l'azione *Reveal* cliccando col tasto destro del mouse sull'immagine.
- Digitare la password (passphrase) precedentemente decrittata col software Mathematica e scegliere il primo algoritmo di cifratura proposto (IDEA).



Fig. 11 Steganalisi mediante il software S Tools

- Se la password è giusta compare il file col messaggio nascosto, salvarlo (cliccare col tasto destro sul file e selezionare l'azione *Save as*) e poi aprirlo..

➔ SE SIETE STATI FORTUNATI AVETE SCOPERTO IL COLPEVOLE,  
ALTRIMENTI PROVATE CON LE ALTRE IMMAGINI..

**BUONA INDAGINE**