

La scimmia cifratrice

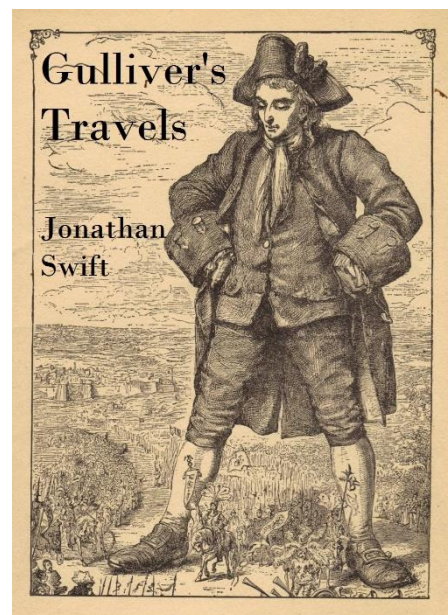
In un laboratorio si vuole realizzare un generatore di simboli, denominato “dispositivo scimmia” che generi quelli prodotti da un algoritmo di cifratura.

Utilizzando un programma dotato di una interfaccia, nel quale è presente un form in cui bisogna autenticarsi, si accede ad una sezione del programma nella quale è possibile inserire la frase da criptare e creare diverse impostazioni di criptazione. La frase inserita viene criptata e salvata in una tabella di una base dati. Contemporaneamente però si attiva un dispositivo (la scimmia) connesso mediante porta seriale, al quale si richiede di generare una sequenza di simboli di pari lunghezza rispetto a quella della frase criptata. Questa sequenza di caratteri rappresenterà la cifratura scimmiottata e sarà anch’essa salvata nella base dati.

L’obiettivo di questo progetto è unire il Test di Turing con il teorema della scimmia instancabile, infatti comparando i risultati del dispositivo scimmia alle frasi criptate, è indistinguibile quale dei due origini da una frase scritta da un essere umano.

Secondo il teorema della scimmia instancabile una scimmia che preme i tasti di una tastiera a caso, per un tempo infinitamente lungo quasi certamente riuscirà a comporre qualsiasi testo prefissato, come ad esempio la “Divina commedia” di Dante.

Anche nella letteratura è presente nei “Viaggi di Gulliver” (1726) di Jonathan Swift, l’idea di produrre qualsiasi testo combinando casualmente le lettere dell’alfabeto.



Questo teorema in matematica si traduce nella funzione: $(1 - \frac{1}{m^k})^n$.

Data una tastiera di tasti m e un testo da riprodurre di n battute, la probabilità di non effettuarlo in tentativi (indipendenti) è $(1 - \frac{1}{m^k})^n$ e il limite $n \rightarrow +\infty$ porta tutta l'espressione a 0 perciò la probabilità di riprodurre un testo fissato se si prova all'infinito è 1.

L'algoritmo di cifratura scelto per il progetto è quello di Vigenère, considerabile come una generalizzazione del cifrario di Cesare. In un cifrario di Cesare, ogni lettera durante la cifratura viene spostata di un certo numero di lettere, per essere sostituita dalla lettera corrispondente.

Nel cifrario di Vigenère invece le lettere vengono spostate di un numero variabile determinato dalle lettere della chiave (In genere Più corta della frase da criptare).

	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
A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

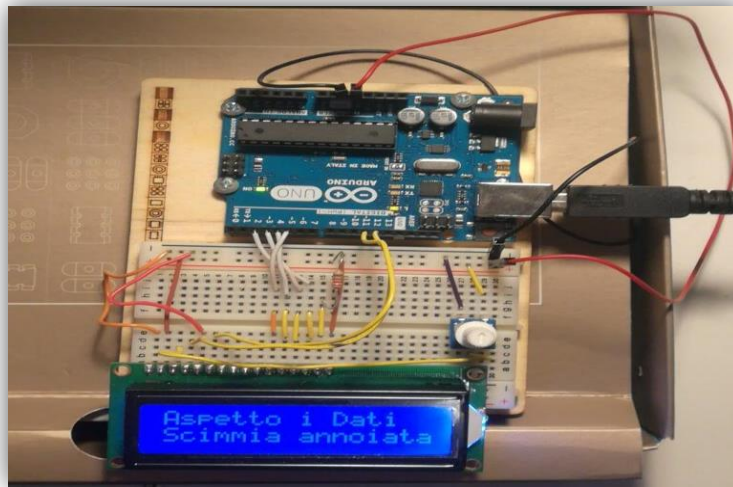
Testo chiaro - ARRIVANOIRINFORZI
 Verme - VERMEVERMEVERMEVE
 Testo cifrato - VVIUZVRFUVDRAWVUM



Il dispositivo “scimmia”

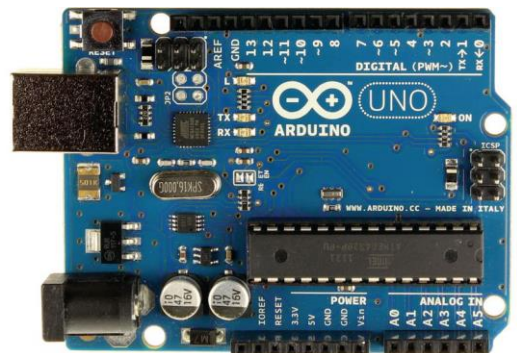
• Il Circuito

Il circuito è stato realizzato fisicamente per testarne il funzionamento:

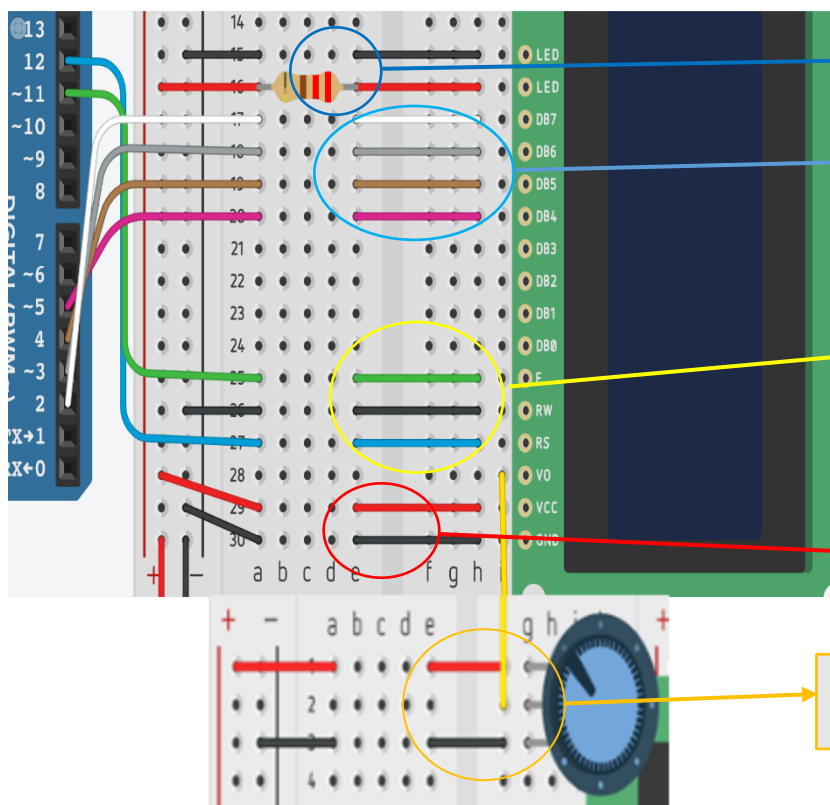
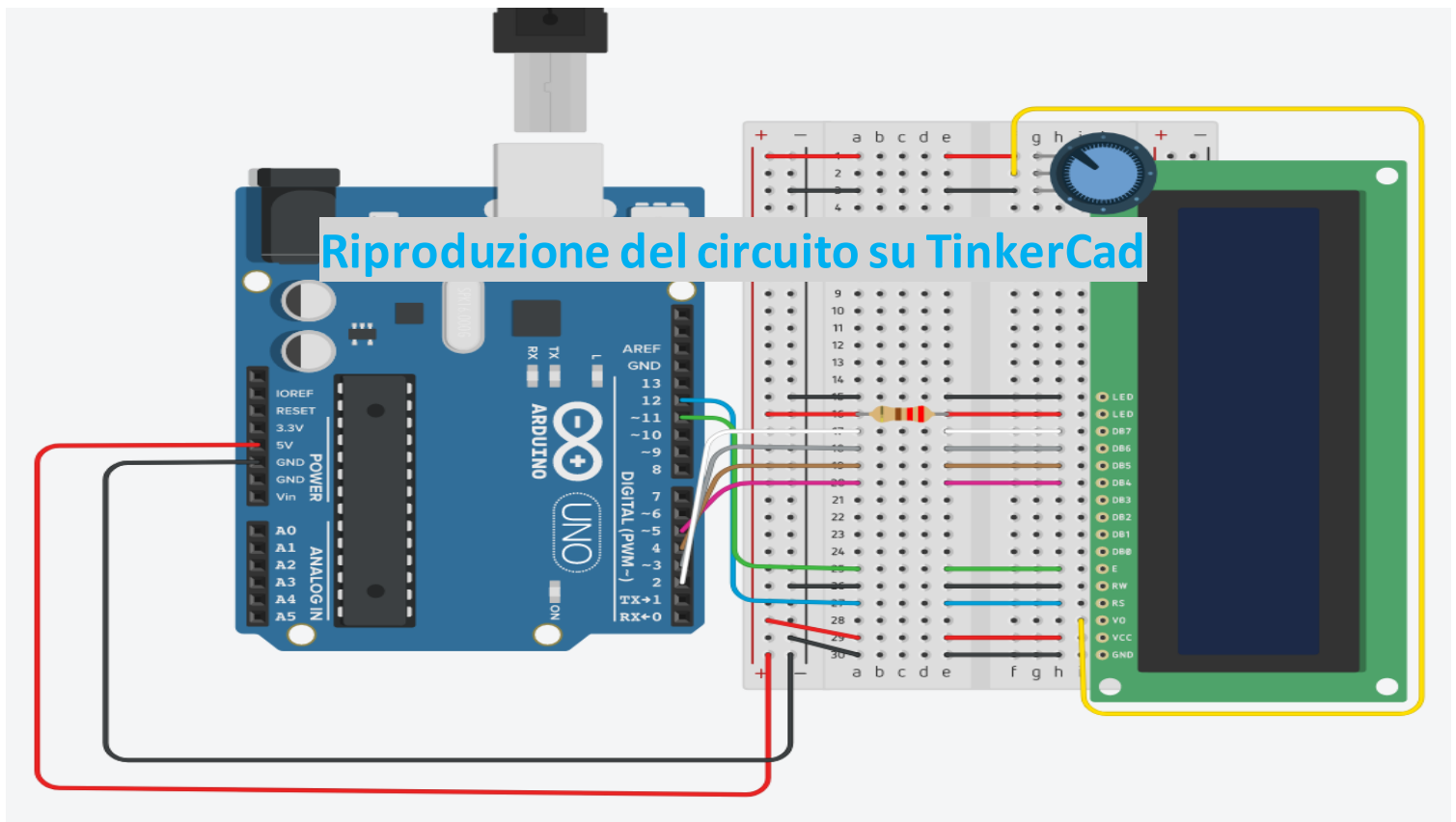


I componenti utilizzati sono:

1. Arduino
2. LCD 16x2
3. Potenziometro
4. Resistenza 220Ω
5. Cavi in Rame (Ponticelli)



Riproduzione del circuito su TinkerCad



Alimentazione del LED per la retroilluminazione

Collegamento dei pin (5-2) sui piedini dei dati (DB4-DB7)

Collegamento dei pin 12-11 sui piedini:

- RS Controllo posizione dei caratteri
- E Comunica all'LCD che riceverà un comando

Il Piedino R/W è collegato a massa (Modalità scrittura)

Alimentazione dell'LCD

Collegamento della tensione in uscita dal potenziometro sul piedino VO (controllo del contrasto dell'LCD)

Lo sketch

Nello sketch ho utilizzato due librerie:

1) **LiquidCrystal** è la libreria inclusa in Arduino che permette la gestione di uno schermo LCD;

2) ***Entropy** è una libreria esterna che permette la creazione di numeri casuali e non pseudo-casuali come il metodo base di Arduino "**random()**";

Creo e inizializzo le variabili:

- 1) Creo l'oggetto lcd indicandogli i pin utilizzati;
- 2) **numeroCar** che conterrà il numero di caratteri ricevuto dalla seriale per la creazione della stringa;
- 3) **monkeyString** ovvero l'array di **char** che conterrà la stringa "scimmiottata";
- 4) **stringStart** - **stringEnd** - **scrollCursor**;

Sono 3 variabili per controllare la stampa dei caratteri nell' LCD.

LCD | Arduino 1.8.13

File Modifica Sketch Strumenti Aiuto



LCD

```
#include <LiquidCrystal.h>
```

```
#include <Entropy.h>
```

```
LiquidCrystal lcd(12,11,5,4,3,2)
```

```
byte numeroCar=0;
```

```
char monkeyString[255]=""; //in
```

```
int stringStart, stringEnd = 0;
```

```
int scrollCursor = 16;
```

*La libreria Entropy

I computer hanno diversi timer che dovrebbero fornire un pattern prevedibile di impulsi a una frequenza fissa (o impostabile).

In realtà l'ampiezza degli impulsi varia. I tempi di salita e di discesa non sono zero. C'è del rumore nei segnali elettrici.

Questo fenomeno è conosciuto come "Jitter" ed è sfruttabile per estrarre entropia, da cui viene il nome della libreria.

Metodo **Void setup()**:

1) Inizio la comunicazione con la seriale;

2) Inizializzo Entropy per la creazione dei numeri casuali:

Questo metodo configura i timer del chip e imposta le strutture interne necessarie per convertire il jitter dei timer hardware in un flusso imparziale di entropia;

3) Inizializza l'interfaccia con l'LCD;

```
void setup() {  
  Serial.begin(115200);  
  Entropy.initialize();  
  lcd.begin(16,2);  
}
```

Metodo **createRandomString()**:

Imposto la fine della stringa nella cella che corrisponde all'ultimo carattere + 1.

Inizia il ciclo di creazione della stringa "scimmiettata":

Il primo metodo utilizzato è **randomSeed()** dove viene inserito come parametro il metodo **Entropy.random()** che restituirà un numero casuale di tipo unsigned long (32 bit).

Il Seed precedentemente generato verrà utilizzato dal metodo **random()** per scegliere un numero casuale tra 33 e 127 saltando i valori corrispondenti al backslash, i doppi apici e il singolo apice, che verrà convertito nel relativo carattere della tabella ASCII.

```
void createRandomString()
{
    monkeyString[numeroCar]='\0';

    for(int i=0;i<numeroCar;i++)
    {
        byte randomValue;
        do
        {
            randomSeed(Entropy.random()); // Vi
            randomValue = random(33, 127);
        }while(randomValue==92 || randomValue
        char letter = randomValue ;
        monkeyString[i]=letter; // Salvo la l
    }
}
```

Metodo **Void loop()**:

Il metodo Loop è il metodo che Arduino continuerà a ripetere fino a che non verrà spento.

Imposto la posizione del cursore dell'LCD con il metodo **.setCursor()** e stampo due messaggi utilizzando il metodo **.print()** per indicare l'attesa del numero dei caratteri dalla seriale.

```
void loop()
{
    stringEnd = 0;
    stringStart = 0;
    scrollCursor = 16;

    lcd.setCursor(0,0);
    lcd.print("Aspetto i Dati"); // Stam
    lcd.setCursor(0,1);
    lcd.print("Scimmia annoiata...");
    delay(1000);
    if(Serial.available())
```

Nel momento in cui ricevo un segnale dalla porta seriale, leggo il messaggio utilizzando il metodo **Serial.parseInt()** che per un periodo di tempo aspetta che arrivi il numero di caratteri, in modo tale da non perdere parte dell'informazione, successivamente lo salva come un numero intero.

I contenuti dell'LCD vengono azzerati con il metodo **.clear()**, se il numero dei caratteri è diverso da 0 viene stampato un messaggio per indicare l'elaborazione della stringa "scimmiettata" e viene richiamato il metodo **createRandomString()**.

```
if(Serial.available())
{
    // Ricevo un segnale dalla porta seriale
    numeroCar=Serial.parseInt(); // Leggo il numero
    lcd.clear();
    lcd.setCursor(0,0);
    if(numeroCar!=0){lcd.print("Scimmia scrive..");}
    createRandomString(); // Richiamo il metodo per
    if(numeroCar>16)
```

Se il numero dei caratteri è maggiore delle celle disponibili nel display, inizia il ciclo di stampa a scorrimento nella riga inferiore della stringa "scimmiettata" creata.

Il metodo della libreria **LiquidCrystal** **.scrollDisplayLeft()** non permette lo scorrimento di una singola riga, ed è quindi necessario simulare lo scorrimento, utilizzando un ciclo di stampa dei caratteri uno alla volta, tenendo conto della posizione del cursore, del primo e dell'ultimo carattere visualizzato.

I caratteri vengono stampati ciclicamente partendo dal valore di **stringStart**.

Se viene raggiunto il numero totale dei caratteri dopo un **delay()**, che blocca l'esecuzione per un tempo in millisecondi inserito come parametro, viene impostato **stringEnd** a un valore impossibile da raggiungere (maggiore della capienza massima di **monkeyString**) e viene interrotto il ciclo con un **break**.

Fuori dal secondo ciclo viene verificato se è stata stampata tutta la stringa e se la condizione è vera, viene interrotto il primo ciclo con un altro **break**.

Attraverso una selezione si decide quali variabili di controllo del display aumentare o diminuire.

Se il cursore non è stato ancora spostato di 16 celle, si diminuisce **scrollCursor** e si aumenta il valore di **stringEnd**, in modo tale da stampare il prossimo carattere nella prossima iterazione.

Nel momento in cui il cursore raggiunge zero, basta aumentare **stringStart** e **stringEnd** a tutte le successive iterazioni, per mantenere l'effetto di scorrimento.

```
if(numeroCar>16)
{
    for(int i=0;i<numeroCar;i--)
    {
        // Ciclo di stampa e scorrimento della stringa

        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Scimmia dice:"); // Stringa da stampare
        lcd.setCursor(scrollCursor,1); // cambio riga

        for(int f=stringStart;f<=stringEnd;f++)
        {
            // stampo tutti i caratteri a partire da stringStart

            lcd.print(monkeyString[f]);
            if(f==numeroCar)
            {
                // Nel momento in cui abbiamo stampato tutti i caratteri
                delay(2000);
                stringEnd=256;
                break;
            }
        }
    }
    if(stringEnd==256)
    {
        // Se abbiamo terminato la stampa della stringa
        delay(1000);
        break;
    }
    delay(500);
}
```

```
if(scrollCursor > 0)
{
    scrollCursor--; // spostamento cursore
    stringEnd++;
}
else
{
    // aumento le variabili di controllo
    stringStart++;
    stringEnd++;
}
```

Se il numero dei caratteri è minore o uguale al numero delle celle dell’LCD, la stringa viene semplicemente stampata usando **.print()**.

Dopo aver visualizzato la stringa nell’LCD, questa viene inviata alla porta seriale con il metodo **.println()**, che la invia come un testo ASCII con alla fine **\r** che indica il termine della riga.

```

else
{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Scimmia dice:");
    lcd.setCursor(0,1);
    lcd.print(monkeyString);
    delay(3000);
}

if(numeroCar!=0)
{
    Serial.println(monkeyString);
}

}

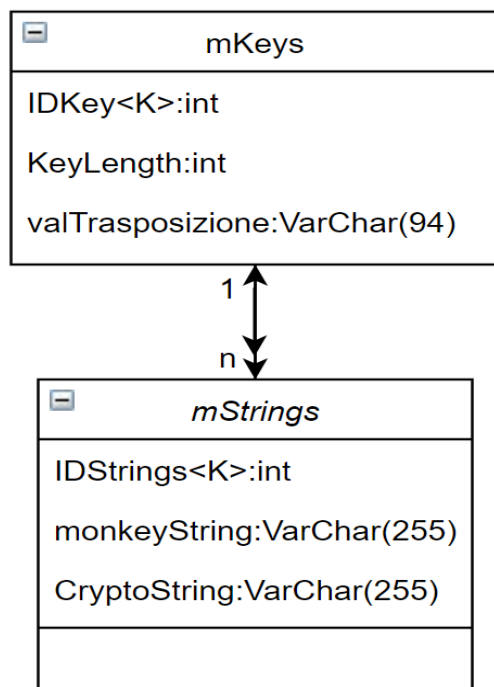
delay(1000);
}

```

La base dati

Prima di creare il programma in C# e il sito web, è necessario creare il database con cui scambieranno i dati le due interfacce.

Modello Concettuale:



La tabella mKeys conterrà le impostazioni di criptazione, mentre la tabella mStrings conterrà la stringa criptata e la stringa “scimmiottata”.

Associazione:

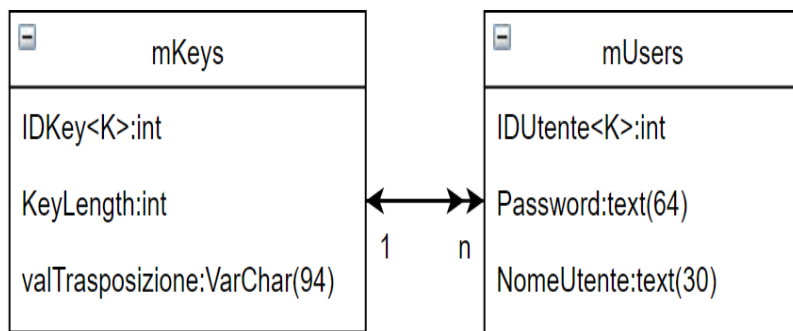
Le impostazioni di criptazione possono essere utilizzate da diverse frasi criptate.

Una frase criptata può utilizzare una sola impostazione.

Modello Logico:

mStrings(**IDStrings**,monkeyString,CryptoStrng,**Key_ID***)

mKeys(**IDKey**,KeyLength,valTrasposizione)



La tabella mUsers conterrà le informazioni per l'autenticazione degli utenti.

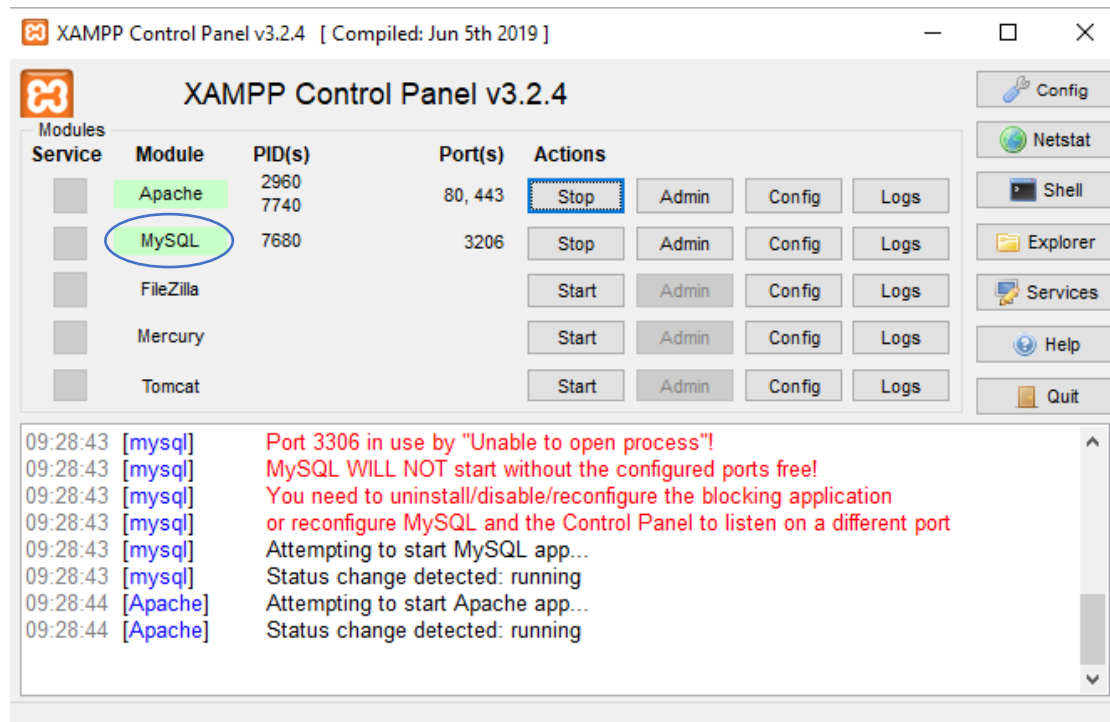
Associazione:

Un utente può creare diverse impostazioni, ed una impostazione associata ad un solo utente.

Modello Logico:

mUsers(IDUtente, NomeUtente, Password)

La base dati è stata creata utilizzando il server MySQL in locale fornito dal programma XAMPP:



Creazione del database e delle tabelle

Creo il database “monkey” e inserisco la tabella mUsers.

La chiave primaria è IDUtente che sarà impostata in **AUTO_INCREMENT** che genera un nuovo numero **int** univoco ogni volta che si inserisce una nuova riga.

NomeUtente e Password sono dei campi di tipo **VARCHAR** che contengono una stringa di lunghezza variabile.

```
CREATE DATABASE monkey;
```

```
CREATE TABLE mUsers(
  IDUtente int PRIMARY KEY AUTO_INCREMENT,
  NomeUtente VARCHAR(30),
  Password VARCHAR(64)
```

Inserisco la tabella **mKeys**:

- **KeyLength**: numero intero che conterrà la lunghezza della chiave di criptazione;
- **Utente_ID** conterrà l'id dell'utente che ha creato l'impostazione;
- **valTrasposizione**: una stringa che contiene i valori di trasposizione della chiave;
- Impostiamo **Utente_ID** come chiave esterna che si riferisce alla chiave primaria della tabella **mUsers (IDUsers)**.

```
CREATE TABLE mKeys(  
IDKey int PRIMARY KEY AUTO_INCREMENT,  
Utente_ID int,  
KeyLength int ,  
valTrasposizione VARCHAR(510),  
FOREIGN KEY(Utente_ID) REFERENCES mUsers(IDUtente)  
)
```

Inserisco la tabella **mStrings**

- **Key_ID**: conterrà l'id delle impostazione di criptazione con cui è stata creata la frase criptata;
- **monkeyString**: la stringa "scimmiottata";
- **CryptoString**: la frase criptata;

```
CREATE TABLE mStrings(  
IDStrings int PRIMARY KEY AUTO_INCREMENT,  
Key_ID int,  
monkeyString VARCHAR(255) ,  
CryptoString VARCHAR(255) ,  
FOREIGN KEY(Key_ID) REFERENCES mKeys(IDKey)  
);
```

Inoltre sono stati creati 2 Utenti del DBMS:

L'utente "monkey" sarà utilizzato dal programma C# che dovrà poter leggere e inserire dati nelle tabelle **mUsers**, **mStrings** e **mKeys**.

```
CREATE USER 'monkey' IDENTIFIED BY 'controller'  
GRANT SELECT,INSERT ON monkey.* TO 'monkey' IDENTIFIED BY 'controller'
```

L'utente "webUser" sarà utilizzato dall'interfaccia web, che dovrà poter leggere i dati presenti nelle tabelle **mKeys** e **mStrings**.

```
CREATE USER 'webUser' IDENTIFIED BY 'web'  
GRANT SELECT ON monkey.* TO 'webUser' IDENTIFIED BY 'web'
```

Il programma C#

Il programma creato, sviluppato nella IDE di **Visual Studio**, usa come base "WindowsFormsApp" un'applicazione basata sugli eventi:

Codice e layout dell'applicazione:

All'inizio del progetto ho creato le principali variabili che ho utilizzato nell'applicazione:

- **currentKey**: La chiave utilizzata per la crittazione più recente;
- **transpVal**: valori di trasposizione della chiave
- **currentVigenereTb**: array bidimensionale di caratteri che conterrà la tabella di Vigenere per la crittazione;
- **phRow**: Caratteri utilizzati per la creazione della tabella di Vigenere;
- **arduinoOutput**: La stringa ricevuta dalla porta seriale;

```
public partial class Form1 : Form
{
    string currentKey;
    int[] transpVal; // Valori di cifratura
    char[,] currentVigenereTb; // tabella di vi
    static string phRow = "!#$%&'()*+,-./012345
    string arduinoOutput; // Conterra` il risul
```

Login dell'utente:

```
private void buttonConfermaLogin_Click(object sender, EventArgs e)
{
    try
    {
        string sql = "SELECT NomeUtente,Password FROM mUsers WHERE NomeUtente='" + textBoxLoginUtente.Text +
            "' AND Password='" + textBoxLoginPassword.Text.Trim().GetHashCode() + "'";
        MySqlDataReader rdr = GestioneMySQL.RequestQuery(sql);
```

Quando si preme su conferma, viene richiamato il metodo **_Click** del controllo Button premuto:

Il codice all'interno di questo metodo è contenuto in un **try{} catch{}** che gestirà le possibili eccezioni che il codice può lanciare (Errori di connessione al database, o errori di RunTime).

Creo la **stringa sql** che contiene la Query con un **SELECT** per controllare se l'utente inserito è presente nel Database.

Creo un'oggetto di tipo **MySqlDataReader**, questo conterrà il risultato del metodo **RequestQuery** della libreria **GestioneMySQL** che utilizza gli strumenti di C# per la gestione dei database MySQL.

Se la query ha restituito una riga, significa che l'utente esiste all'interno del database:

Viene permesso all'utente di accedere all'interfaccia per la comunicazione con il dispositivo scimmia, la criptazione e il salvataggio delle frasi nel database.

Viene richiamato il metodo che inizierà la comunicazione del programma con la porta seriale collegata ad Arduino.

```
if (rdr != null && rdr.HasRows)
{
    buttonConfermaLogin.Enabled = false;
    buttonRegistrazione.Enabled = false;
    MessageBox.Show("Login completato");
    buttonConfermaLogin.Enabled = true;
    buttonRegistrazione.Enabled = true;
    panelLogin.Visible = false;
    panelLogin.Enabled = false;
    rdr.Close();
    GestioneMySQL.ChiudiConnessione();
    checkDbCryptoSettings();
    Invoke(new EventHandler(btnConnetti_Click));
}
```

Registrazione dell'utente:

```
if ((textBoxRegConfermaPassword.Text.Equals(textBoxRegPassword.Text)) && (textBoxRegConfermaPassword.Text != ""))
{
    string sql = "SELECT NomeUtente FROM monkey.mUsers WHERE NomeUtente='" + textBoxRegUtente.Text.ToString() + "'";
    try
    {
        MySqlDataReader rdr = GestioneMySQL.RequestQuery(sql);
        if (rdr != null && rdr.HasRows)
        {
            MessageBox.Show("Questo nome utente e` gia` in uso");
            GestioneMySQL.ChiudiConnessione();
        }
        else
        {
            GestioneMySQL.ChiudiConnessione();
            completeReg();
        }
        textBoxRegUtente.Text = "";
        textBoxRegPassword.Text = "";
    }
}
```

Come nel login, controlliamo con una Query se esiste già un utente con lo stesso nome, se non esiste alcun utente si procede con la registrazione nel metodo **completeReg()**

La query con un INSERT inserirà nella tabella **mUsers** il nuovo utente, se l'operazione ha successo si potrà accedere dalla pagina di login con l'utente appena creato.

```
private bool completeReg()
{
    try
    {
        StringBuilder sql = new StringBuilder();
        sql.AppendLine("INSERT monkey.mUsers ");
        sql.AppendLine("(NomeUtente,Password) VALUES('" + tex

        if (GestioneMySQL.InsertValues(sql.ToString()))
        {
            buttonConfermaReg.Enabled = false;
            MessageBox.Show("Registrazione completata");
            buttonConfermaReg.Enabled = true;

            return true;
        }
    }
}
```

Pagina principale:

The diagram shows three rectangular input fields stacked vertically. The top field has a red border, the middle field has a black border, and the bottom field has a yellow border.

Questa interfaccia permette:

- La scrittura della frase da criptare;
- La scelta di un'impostazione di criptazione già creata e memorizzata nel database;
- Il bottone in fondo alla pagina fa apparire la scheda per la creazione di una nuova impostazione di criptazione.



```

if (!textBoxNewKey.Text.Contains(" "))
{
    try
    {
        int c = 0;
        currentKey = textBoxNewKey.Text.Trim();
        findTranspVal();
    }
}

```

Quando si preme il bottone di conferma vengono ricavati i valori di trasposizione della chiave inserita utilizzando il metodo richiamato **findTranspVal()**.

Per ogni carattere con il **for** viene trovato il rispettivo valore di trasposizione utilizzando il metodo **IndexOf()** per cercare nella stringa **phRow** la cella in cui è presente lo stesso carattere.

```

private void findTranspVal()
{
    string key = currentKey;
    int keyCounter = 0;
    transpVal = new int[key.Length];

    for (int counter = 0; counter < key.Length; counter++, keyCounter++)
    {
        if (keyCounter >= key.Length)
        {
            keyCounter = 0;
        }
        transpVal[keyCounter] = phRow.IndexOf(key[keyCounter]);
    }
}

```

Viene scritta la stringa contenente i valori di trasposizione separati da spazi utilizzando un ciclo **while**.

```

findTranspVal();
string ValoriTrasposizione = "";
ValoriTrasposizione = transpVal[c].ToString();
c++;
while (c < transpVal.Length)
{
    ValoriTrasposizione = ValoriTrasposizione + ' ';
    ValoriTrasposizione = ValoriTrasposizione + transpVal[c].ToString();
    c++;
}

```

Controllo nel database con un **SELECT** che non esista un'impostazione uguale per l'utente selezionato.

```

string sqlCheck =
"SELECT Utente_ID,KeyLength,valTrasposizione " +
"FROM monkey.mKeys " +
"WHERE KeyLength=" + currentKey.Length +
" AND valTrasposizione='" + ValoriTrasposizione +
"'AND Utente_ID="+ UserId +
"';";
MySQLDataReader rdr = GestioneMySQL.RequestQuery(sqlCheck);

if (rdr != null && rdr.HasRows == true)
{
    MessageBox.Show("Questa impostazione esiste già!");
    rdr.Close();
    GestioneMySQL.ChiudiConnessione();
}
else
{
    rdr.Close();
    StringBuilder sql = new StringBuilder();
    sql.AppendLine("INSERT monkey.mkeys ");
    sql.AppendLine(" (KeyLength,valTrasposizione) VALUES (" + currentKey.Length);
    sql.AppendLine(", '" + ValoriTrasposizione + "')");
    sql.AppendLine("INSERT monkey.mkeys ");
    sql.AppendLine(" (Utente_ID,KeyLength,valTrasposizione) VALUES (" + UserId + "," + currentKey.Length);
    sql.AppendLine(", '" + ValoriTrasposizione + "')");
}
}

```

Inserisco nel database con un **INSERT** le impostazioni create dall'utente selezionato.

La scrittura della frase criptata



```
private void buttonSendSaveDb_Click(object sender, EventArgs e)
{
    currentKey = comboBoxKey.Text;
    Invoke(new EventHandler(btnScrivi_Click));
}
```

Dopo aver premuto il pulsante, verrà richiamato il metodo che invierà nella porta seriale la lunghezza della stringa che dovrà scrivere il “dispositivo scimmia”.

```
private void btnScrivi_Click(object sender, EventArgs e)
{
    textBoxFrase.Text.Trim();

    if (!textBoxFrase.Text.Contains(" "))
    {
        if (serialPort.IsOpen && textBoxFrase.Text.Length > 0 && textBoxFrase.Text != " ")
        {
            serialPort.WriteLine(textBoxFrase.Text.Length.ToString());
        }
        else
        {
            MessageBox.Show("Inserire un valore valido e verificare che la scimmia sia connessa");
        }
    }
}
```

Quando il “dispositivo scimmia” avrà inviato la stringa scimmiettata, il metodo **SerialPort.DataReceived()** gestirà l’evento di ricezione dei dati dalla porta seriale, e richiamerà **ArduinoRequestAndSaveToDB()** per salvare le stringhe nel database.

```
private void serialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        arduinoOutput = serialPort.ReadLine().Trim();
        Invoke(new EventHandler(ArduinoRequestAndSaveToDB));
    }
    catch
    {
    }
}
```

Se non è già stata creata si procederà con la creazione della tabella di Vigenere utilizzando il metodo **CreateVigenereTable()** che prenderà come parametro la stringa contenente i caratteri che dovrà disporre all'interno della tabella.

```
private void ArduinoRequestAndSaveToDB(object sender, EventArgs e)
{
    if (arduinoOutput != null && arduinoOutput != "\\0" && currentKey != null && combo
    {
        if (currentVigenereTb == null)
        {
            CreateVigenereTable(phRow);
        }
    }
}
```

Viene inizializzata la variabile **modifier**, **CharCursor** e **currentVigenereTb**.

All'interno del primo ciclo la variabile **CharCursor** prenderà come valore iniziale il valore di **modifier**, quest'ultimo verrà incrementato ad ogni iterazione in modo tale che all'inizio di ogni ripetizione del secondo ciclo **for**, si cominci a scrivere i caratteri della riga attuale a partire dal carattere precedente + 1.

Nel momento in cui si raggiunge l'ultimo carattere disponibile, si azzerà **CharCursor** ripartendo dall'inizio di **TrValues**

```
1 riferimento
private void CreateVigenereTable(string TrValues)
{
    int modifier = 0;
    int CharCursor = modifier;
    currentVigenereTb = new char[TrValues.Length, TrValues.Length];
    for (int y = 0; y < TrValues.Length; y++)
    {
        CharCursor = modifier;
        for (int x = 0; x < TrValues.Length; x++)
        {
            if (CharCursor < TrValues.Length)
            {
                currentVigenereTb[y, x] = TrValues[CharCursor];
                CharCursor++;
            }
            else
            {
                x--;
                CharCursor = 0;
            }
        }
        modifier++;
    }
}
```

```
reverseTranspVal(comboBoxValTransp.Text);
string cryptutedString = CryptString(currentVigenereTb, currentKey, textBoxFrase.Text).ToString();
string monkeyString = arduinoOutput;
txtOutput.AppendText(arduinoOutput);
```

Dopo la creazione della tabella, verrà richiamato **reverseTranspVal()** che inserirà all'interno dell'array **TranspVal**, dichiarato all'inizio del codice, i valori di trasposizione della chiave selezionata. Questo passaggio è necessario per poter leggere correttamente i valori ricevuti dal database, che sono separati tra loro dagli spazi.

Ora che abbiamo i valori di trasposizione si può procedere con la criptazione della frase inserita, richiamando **CryptString()** che prende come parametri la tabella di Vigenere, la chiave e la frase da criptare. Il risultato verrà salvato nella stringa temporanea **cryptutedString**.

A ogni iterazione del ciclo **for**, verrà aggiunto alla stringa criptata il carattere trasposto presente nella tabella.

Quando si raggiunge l'ultimo carattere della chiave, si resetta **keyCounter**, continuando la criptazione ripartendo dal primo carattere.

```
private string CryptString(char[,] table, string key, string sourceString)
{
    int keyCounter = 0;
    string cripted = "";

    for (int sourceCounter = 0; sourceCounter < sourceString.Length; sourceCounter++, keyCounter++)
    {
        if (keyCounter >= key.Length)
        {
            keyCounter = 0;
        }
        cripted = .. cripted+table[phRow.IndexOf(key[keyCounter]), phRow.IndexOf(sourceString[sourceCounter])];
    }
    return cripted;
}
```

Prima di inserire i dati all'interno del database, troviamo l'ID delle impostazioni selezionate richiedendo con una query l'ID della riga con le impostazioni selezionate.

Inseriamo con un **INSERT** la nuova riga nella tabella **mStrings** e comunichiamo all'utente il successo dell'operazione.

```
try
{
    StringBuilder sql = new StringBuilder();
    MySqlDataReader rdr = GestioneMySql.RequestQuery(
        "SELECT IDKey " +
        "FROM monkey.mkeys " +
        "WHERE KeyLength=" + Int32.Parse(comboBoxKey.Text) +
        " AND valTrasposizione='" + comboBoxValTransp.Text + "';");
    rdr.Read();

    int idKeySettings = Convert.ToInt32(rdr[0]);
    sql.AppendLine("INSERT monkey.mstrings ");
    sql.AppendLine(" (Key_ID,monkeyString,CryptoString) VALUES (");
    sql.AppendLine(idKeySettings + "," + monkeyString + "," + cryptedString + ")");
    rdr.Close();
    if (GestioneMySql.InsertValues(sql.ToString()))
    {
        MessageBox.Show("Dati inseriti con successo!");
    }
    else
    {
        MessageBox.Show("Errore nell'inserimento dati");
    }
}
```


L'interfaccia web

Il Database remoto, e' accessibile mediante una interfaccia web, realizzata in php. Nel sito e' possibile visualizzare una descrizione del progetto e vari dati statistici estratti dai dati contenuti nella base dati.

[Homepage](#) [Le statistiche](#)

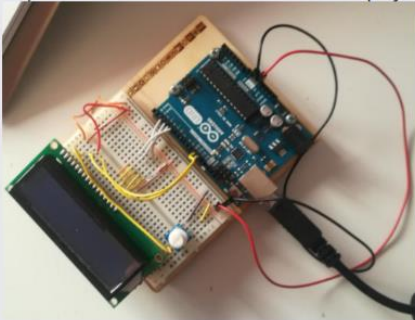
LA SCIMMIA CIFRATRICE

DATI RECENTI

Frase salvate:	68
Caratteri stringa minore:	1
Caratteri stringa maggiore:	120
Lunghezza media:	17

Il Dispositivo Scimmia

Il dispositivo Scimmia e' stato realizzato con Arduino, il suo obiettivo e' di rispondere ad una richiesta sulla porta seriale generando una sequenza di simboli casuali della lunghezza richiesta. La sequenza viene inoltre visualizzata su un display LCD e trasmessa via seriale



Richiamiamo l'intestazione del sito e le impostazioni di connessione al database.

I Dati ricevuti dal database sono disposti in un tag `<aside>`.

Con un `SELECT` richiediamo la quantità di frasi salvate all'interno della tabella `mStrings` contando il numero di id con un `COUNT()`.

```
<?php
include ("ints.php");
include ("dbConnection.php");
?>
<div id="main-content">
  <aside>
    <h1>Dati recenti</h1>
    <hr />
    <h3>Frase salvate:
    <a >
    <p>
    <?php
      $sql="SELECT COUNT(IdStrings) AS Stringnum FROM monkey.mStrings ";
      $result=mysqli_query($connessione, $sql);
      if(mysqli_num_rows($result)>0)
      {
        while($row = mysqli_fetch_assoc($result))
          printf("%s.$row['Stringnum'].");
      }
    ?>
    </p>
    </a>
  </h3>
```

Con un **SELECT** richiediamo la frase piu` corta contenuta nel database mStrings con l'istruzione **MIN()** in cui inseriamo **LENGTH()**.

Per richiedere la frase più corta utilizziamo la precedente query ma al posto di **MIN()** utilizziamo **MAX()**.

Riceviamo la lunghezza media delle frasi inserendo **AVG()** al posto di **MAX()**.

```
<hr />
<h3>Caratteri stringa minore:
<a>
  <p>
    <?php
      $sql="SELECT MIN(LENGTH(CryptoString)) AS MinString FROM monkey.mStrings;";
      $result=mysqli_query($connessione, $sql);
      if(mysqli_num_rows($result)>0)
      {
        while($row = mysqli_fetch_assoc($result))
          printf("%.5row['MinString']")."";
      }
    ?>
  </p>
</a>
</h3>
<hr />
```

```
<hr />
<h3>Caratteri stringa maggiore:
<a>
  <p>
    <?php
      $sql="SELECT MAX(LENGTH(CryptoString)) AS MaxString FROM monkey.mStrings;";
      $result=mysqli_query($connessione, $sql);
      if(mysqli_num_rows($result)>0)
      {
        while($row = mysqli_fetch_assoc($result))
          printf("%.5row['MaxString']")."";
      }
    ?>
  </p>
</a>
</h3>
<hr />
```

```
<hr />
<h3> Lunghezza media:
<a>
  <p>
    <?php
      $sql="SELECT AVG(LENGTH(CryptoString)) AS AvgString FROM monkey.mStrings;";
      $result=mysqli_query($connessione, $sql);
      if(mysqli_num_rows($result)>0)
      {
        while($row = mysqli_fetch_assoc($result))
          printf("%.round($row['AvgString'])."";
      }
    ?>
  </p>
</a>
</h3>
<hr />
```

Le statistiche

Homepage Le statistiche

LA SCIMMIA CIFRATRICE

DATI STATISTICI

Lunghezza chiave	Simboli al posto giusto	Media Simboli giusti	Frase sbagliate
1	1	4.1	60%
3	1	0.2	81.8%
4	0	0	100%
8	2	16.2	30%
9	0	1.3	33.3%
10	1	0.5	50%
17	0	3.3	25%
50	6	10.2	47.1%
115	1	113	57%

In questa pagina vogliamo trovare 3 dati statistici in base alla lunghezza della chiave:

- 1) Quanti simboli mediamente sono in comune tra le vere cifrature e quelle false generate dalla scimmia;
- 2) Quanti simboli mediamente vengono indovinati dalla scimmia (giusti ed al posto giusto);
- 3) La percentuale di cifrature scimmiettate che sono del tutto sbagliate (neanche un simbolo presente nella vera cifratura);

Per realizzare la tabella nella pagina utilizziamo il tag `<table>`

```
<th><h3>Simboli al posto giusto</h3></th>
<th><h3>Simboli giusti</h3></th>
<th><h3>Frase sbagliate</h3></th>
</tr>
<?php
$sameSymbol=0;
$NoSymbolMatching=0;
$cSymbol=0;
$keyLength=0;
$sameSymbolPosition=0;
$sql="SELECT KeyLength, cryptoString, monkeyString FROM monkey.mStrings, monkey.mKeys WHERE mKeys.IDKey=mStrings.Key_ID ORDER BY KeyLength";
$result=mysqli_query($connessione, $sql);
if(mysqli_num_rows($result)>0)
```

Creiamo e inizializziamo le variabili che verranno utilizzate per:

- **sameSymbol** : Contare quanti simboli giusti vengono trovati;
- **NoSymbolMatching** : Contare le frasi completamente sbagliate;
- **cSymbol** : Servirà per controllare se la frase precedentemente analizzata era sbagliata;
- **SameSymbolPosition** : Contare i simboli giusti e al posto giusto;
- **KeyLength** : Conterrà la lunghezza della chiave utilizzata per le stringhe analizzate;

Oltre a queste variabili, creiamo una stringa **sql** per richiedere la lunghezza della chiave dalla tabella **mKeys** e le due frasi dalla tabella **mStrings**, ordinate secondo la lunghezza della chiave.

Inizia il ciclo **while** che continua fino a che ci sono righe da leggere nella tabella ricevuta. A inizio ciclo vengono incrementate le variabili **currentRow** che conterrà le chiavi della stessa lunghezza e **rowCount** che terrà conto di tutte le righe.

Inoltre, alla prima iterazione viene assegnata la variabile **KeyLength** alla lunghezza di chiave attuale.

A ogni ripetizione si controlla se la lunghezza di chiave è cambiata e se questo è vero, vengono stampate in una riga della tabella i dati utilizzando i tag **html** `<tr>` e `<th>`. Dopo aver azzerato le varie variabili di controllo si assegna **keyLength** alla lunghezza della chiave attuale.

```
$result=mysqli_query($connessione, $sql);
if(mysqli_num_rows($result)>0)
{
    $rowCount=0;
    $currentRow=0;

    while($row = mysqli_fetch_assoc($result))
    {
        $currentRow++;
        $rowCount++;
        if($keyLength==0)
        {
            $keyLength=$row['KeyLength'];
        }
    }
}
```

```
if($keyLength!=$row['KeyLength'])
{
    echo"<tr>";
    echo "<th><h3> ".$keyLength."</h3></th>";
    printf("<th><h3> ".$sameSymbolPosition."</h3></th>");
    printf("<th><h3> ".round(($sameSymbol)/$currentRow,1)."</h3></th>");
    echo"<th><h3> ".round(($NoSymbolMatching/$currentRow)*(100),1)."%</h3></th>";
    echo"</tr>";
    $sameSymbolPosition=0;
    $NoSymbolMatching=0;
    $sameSymbol=0;
    $cSymbol=0;
    $currentRow=0;
    $keyLength=$row['KeyLength'];
}
```

Viene impostato **cSymbol** al valore di **sameSymbol** e vengono salvate le due stringhe della riga attuale.

Vengono inizializzate le stringhe **cS** e **mS** che serviranno per salvare i caratteri delle rispettive stringhe senza ripetizioni.

A ogni iterazione controlliamo se i caratteri presenti nella stessa posizione sono uguali, se sì, incrementiamo la variabile **sameSymbolPosition**.

In questo secondo ciclo, dopo aver controllato che lo stesso carattere non sia già presente utilizzando il metodo **strpos()** con le stringhe **mS** e **cS** viene salvato il carattere attuale.

A ogni ripetizione del terzo ciclo quando viene trovato un simbolo corrispondente si incrementa **sameSymbol**.

Compara **cSymbol** con **sameSymbol** e se il valore non è cambiato significa che la frase analizzata precedentemente era totalmente sbagliata.

Alla fine del ciclo, viene stampata la riga se questa è l'ultima della tabella.

```
$cSymbol=$sameSymbol;  
$cryptoString=$row['cryptoString'];  
$monkeyString=$row['monkeyString'];  
$cS="";  
$mS="";
```

```
for($i=0;$i<strlen($monkeyString);$i++)  
{  
    if($monkeyString[$i]==$cryptoString[$i])  
    {  
        $sameSymbolPosition++;  
    }  
}  
for($i=0;$i<strlen($monkeyString);$i++)  
{  
    if(!strpos($mS,$monkeyString[$i],0))  
    {  
        $mS[$i]=$monkeyString[$i];  
    }  
    if(!strpos($cS,$cryptoString[$i],0))  
    {  
        $cS[$i]=$cryptoString[$i];  
    }  
}  
for($c=0; $c<strlen($cS);$c++)  
{  
    if(strpos($mS,$cS[$c],0))  
    {  
        $sameSymbol++;  
    }  
}  
if($cSymbol==$sameSymbol)  
{  
    $NoSymbolMatching++;  
}  
if(mysqli_num_rows($result)==$rowCounter)  
{  
    echo "<th><h3>".$row['KeyLength']. "</h3></th>";  
    printf("<th><h3>".$sameSymbolPosition. "</h3></th>");  
    printf("<th><h3>".round(($sameSymbol)/$currentRow,1). "</h3></th>");  
    echo "<th><h3>".round((($NoSymbolMatching/$currentRow)*(100),1). "%</h3></th>";  
    $sameSymbolPosition=0;  
    $NoSymbolMatching=0;  
    $sameSymbol=0;  
    $cSymbol=0;  
}
```

La scimmia cifratrice

Gabriele Esu

5D

2021