

Implementazione del cifrario di Leoni in Haskell

Lorenzo LEONI

Università degli studi di Bergamo, Dipartimento di Ingegneria Gestionale,
dell'Informazione e della Produzione

2 maggio 2023

1 Introduzione

L'algoritmo di Leoni prevede l'utilizzo combinato e iterato della versione modificata del cifrario di Cesare e dell'algoritmo di Vigenère. Esso risulta essere più complesso e pertanto violabile meno facilmente rispetto ai cifrari precedentemente citati per via della sua componente aleatoria e iterativa. Per ulteriori dettagli in merito al funzionamento dell'algoritmo si invita a consultare la documentazione di iCipher, sezione 4:

<https://github.com/lamferzon/iCipher/blob/main/documentazione/documentazione.pdf>

2 Descrizione delle funzioni

Le funzione del software possono essere raggruppate in:

- funzioni per il cifrario di Cesare modificato;
- funzioni per il cifrario di Vigenère;
- funzioni per il cifrario di Leoni;
- funzioni per l'avvio dell'applicazione.

2.1 Funzioni per il cifrario di Cesare modificato

Di seguito sono riportate e descritte le funzioni necessarie per l'implementazione della versione modificata dell'algoritmo di Cesare:

- **getLag**: decodifica l'ultimo carattere della stringa da decriptare in modo tale da risalire al *lag* che è stato utilizzato per la sua cifratura. La funzione **ord** appartenente alla libreria **Data.Char** restituisce la posizione in tabella ASCII del carattere che riceve come argomento;
- **checkP** e **checkM**: verificano che lo spostamento di un carattere in tabella ASCII avvenga nell'intervallo definito dalle costanti **start** (= 40) ed **end** (= 127);
- **shiftCaesar**: effettua lo spostamento di **lag** posizioni in tabella ASCII del carattere **char** che riceve come argomento. Se **mod** è maggiore di zero, allora lo spostamento è in avanti, altrimenti è all'indietro. La funzione **chr**, invece, converte un numero intero in un carattere;
- **encryptsCaesar**: funzione ricorsiva avente il compito di invocare lo spostamento in avanti (**mod** di **shiftCaesar** uguale a 1) di **lag** posizioni di ogni carattere costituente la stringa **x:xs**. Una volta raggiunta la coda, a essa viene concatenata la codifica dello spostamento;

- **decryptsCaesar**: opera come **encryptsCaesar** con la differenza che invoca lo spostamento all'indietro (mod di **shiftCaesar** uguale a -1). Inoltre, rimuove l'ultimo carattere della stringa **x:xs** poiché è quello che viene utilizzato per risalire al **lag**, quindi non appartiene alla parola da decifrare.

2.2 Funzioni per il cifrario di Vigenère

Le funzioni che realizzano il cifrario di Vigenère sono:

- **generateKeyIt** e **generateKey**: la prima concatena iterativamente la chiave di cifratura **baseKey** finché non ottiene una stringa di lunghezza uguale o superiore a quella della parola da criptare, mentre la seconda prende i primi **len** caratteri della stringa risultante dall'esecuzione di **generateKeyIt**, cosicché la sua lunghezza coincida con quella della stringa da cifrare, ossia **s**;
- **shiftVig**: opera come **shiftCaesar** con la differenza che il carattere **char1** viene spostato di un numero di posizioni coincidente con la posizione occupata da **char2** in tabella ASCII;
- **encryptsVig** e **decryptsVig**: la funzione **shiftVig** viene applicata a ogni coppia che si ottiene accostando l'*i*-esimo carattere della stringa da criptare **xs** all'*i*-esimo carattere della chiave **ys** a essa corrispondente.

2.3 Funzioni per il cifrario di Leoni

Di seguito sono riportate e descritte le funzioni necessarie per l'implementazione dell'algoritmo di Leoni:

- **removeLastChar**: restituisce una stringa senza la sua coda. Essa serve per rimuovere il carattere necessario per capire di quanto è stato effettuato lo spostamento con il cifrario di Cesare modificato, cosicché possa essere determinata correttamente la chiave per decifrare con l'algoritmo di Vigenère;
- **encrypts** e **decrypts**: la prima applica **encryptsCaesar** all'output di **encryptsVig**, mentre la seconda **decryptsVig** all'output di **decryptsCaesar**. Esse implementano la natura *combinata* dell'algoritmo;
- **repUp** e **repDown**: funzioni ricorsive aventi il compito di ripetere **num** volte rispettivamente **encrypts** e **decrypts**. Esse realizzano la componente *iterativa* del cifrario.

2.4 Funzioni per l'avvio dell'applicazione

Le funzioni necessarie per l'avvio dell'applicazione sono:

- **startApp**: avvia la cifratura o decifrazione con l'algoritmo di Leoni in funzione del valore assunto dalla variabile **choice**. Quest'ultima viene impostata dall'utente a seconda dell'operazione che desidera venga eseguita dall'applicazione sulla stringa **string** che egli fornisce in input;
- **main**: si occupa dell'interazione I/O con l'utente. La funzione **randomRIO (minLag, maxLag)** del pacchetto **System.Random** genera un numero intero casuale tra **minLag** ($= 8$) e **maxLag** ($= 25$); dal momento in cui essa restituisce un oggetto di tipo **IO**, è necessario effettuare il casting esplicito a **Integer** tramite il comando **: Int**.

3 Repository GitHub

Il codice sorgente dell'applicazione è disponibile per il download al seguente repository:

<https://github.com/lamferzon/Leoni-cipher-in-Haskell>

```
** Leoni cipher **
Copyright 2023 Lorenzo Leoni (UniBG)

PN: use only fonts from '(' (pos. 40 in ASCII table) to 'HOME' (pos. 127 in ASCII table)

Insert a word or a phrase: _F3rrar1+46_
Insert a key: ULIVET011
Insert the number of iterations: 50

What do you wanto to do?
1. Encrypting
2. Decrypting
Your choice: 1

Result: Y6=Fd)H[U.~i8b^l?N9b3z5Z=n},N:d`VfETw:AN3R7t+j~H8^N`KZ]j0z1J1z
```

(a)

```
** Leoni cipher **
Copyright 2023 Lorenzo Leoni (UniBG)

PN: use only fonts from '(' (pos. 40 in ASCII table) to 'HOME' (pos. 127 in ASCII table)

Insert a word or a phrase: Y6=Fd)H[U.~i8b^l?N9b3z5Z=n},N:d`VfETw:AN3R7t+j~H8^N`KZ]j0z1J1z
Insert a key: ULIVET011
Insert the number of iterations: 50

What do you wanto to do?
1. Encrypting
2. Decrypting
Your choice: 2

Result: _F3rrar1+46_
```

(b)

Figura 1: un paio di screenshot dell'applicazione in esecuzione.