UNIVERSITÀ DEGLI STUDI DI FIRENZE
Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

Corso di Tecniche Avanzate di Programmazione

CIPHERS: IMPLEMENTAZIONE DI ALCUNI CIFRARI STORICI

MARCO BURACCHI

Prof. Lorenzo Bettini

Anno Accademico 2016-2017

INDICE

1

CENNI PRELIMINARI

INTRODUZIONE

In questo progetto sono stati implementati cinque storici cifrari a chiave simmetrica. I cifrari implementati sono:

- Shift cipher (altrimenti noto come cifrario "Giulio Cesare")
- Vigenere cipher (l'evoluzione del cifrario di Giulio Cesare)
- Substitution cipher
- Affine cipher
- OneTimePad cipher (il cifrario perfetto)

STRUTTURA PROGETTO

Le cinque classi principali che implementano i cinque cifrari sono *Shift*, *Vigenere*, *Substitution*, *Affine* e *OneTimePad*.

L'interfaccia *Parser* definisce i metodi che deve implementare la classe che si occuperà di gestire le stringhe rappresentanti i messaggi, le chiavi e quant'altro all'interno del progetto.

La classe *InputManager* fornisce un'implementazione di tale interfaccia mentre la classe *Constants* contiene delle variabili condivise.

Infine la classe *Main* contiene un piccolo main utile a spiegare le varie funzionalità del programma.

Per quanto riguarda i test, sono presenti nove classi di unit-testing (una per ogni suddetta classe) e cinque classi di integration-testing per controllare l'effettivo funzionamento dell'interazione tra ognuna delle cinque classi che implementano i cifrari e il parser *InputManager*.

2

TESTING

UNIT TESTING

Per tutti i test viene supposto che le stringhe da cifrare siano state processate dal parser e quindi che contengano solamente caratteri effettivamente cifrabili. Per eliminare tale dipendenza da classi concrete, nei vari unit-test un'istanza del parser viene simulata tramite *mocking* e *stubbing* dei metodi. Tale metodo verrà approfondito più avanti.

Andiamo a vedere nel dettaglio i test con i quali sono state implementate le varie classi.

Shift cipher

Lo *shift cipher* è il cifrario più semplice. L'unica cosa che fa è spostare la lettere da cifrare di un certo numero di posizioni avanti nell'alfabeto. Tale numero di posizioni è la chiave del cifrario. Per decifrare il messaggio basterà semplicemente tornare indietro del numero di posizioni indicato dalla chiave. Se per esempio il messaggio è *ciao* e la chiave è 3, il messaggio cifrato sarà *fldr*.

Le uniche due funzionalità richieste a questo cifrario sono quelle di cifratura e decifratura. Tali funzionalità sono state testate nel caso di uno spostamento di zero posizioni (non ottenendo una cifratura), di una posizione, di ventisei posizioni, di ventisette posizioni e di un numero negativo di posizioni(-1). Come caso limite è stata testata anche la cifratura della stringa vuota (il numero di posizioni è ininfluente dato che la cifratura terminerà immediatamente).

Testando anche la decifratura di tutti questi casi sono state coperte le possibili modalità di cifratura/decifratura di una stringa.

Vigenere cipher

Il *cifrario di Vigenère* è una variante dello shift cipher che invece di utilizzare come chiave un numero fisso, utilizza una parola che viene ripetuta tante volte fino al raggiungimento della lunghezza del messaggio che si vuole cifrare. Ogni lettera verrà poi traslata del numero di posizioni corrispondenti alla posizione nell'alfabeto della corrispondente lettera della chiave.

Se per esempio abbiamo un certo messaggio P da cifrare con una chiave K, allora come prima cosa verrà ripetuta K tante volte fino a che la sua lunghezza non raggiunge quella di P e poi ogni i-esima lettera del messaggio P[i] sarà cifrata in P[i]+K[i] modulo 26.

Cifrare il messaggio *testmessage* con la chiave *test* comporterà tre ripetizioni della chiave (che quindi diventerà *testtesttest*). Conseguentemente la prima lettera del messaggio (*t*) verrà spostata di venti posizioni (verrà cifrata con la prima lettera della chiave che è *t* ed è la ventesima lettera dell'alfabeto), la seconda di cinque e così via.

I comportamenti da testare sono la cifratura, la decifratura e il prolungamento della chiave.

Cifratura e decifratura sono testati con stringhe di lunghezza zero, uno e undici caratteri (quest'ultima scelta è dovuta esclusivamente al fatto di voler cifrare una stringa con un senso compiuto che in questo caso è *testmessage*).

L'estensione della chiave viene testata fornendo la stringa *test* come chiave, facendo cifrare il messaggio *testmessage* e verificando che sia stata trasformata in *testtesttest*.

Sono presenti anche tre test che controllano l'inserimento di una chiave illegale. Questi test sono stati inseriti per chiarire meglio il comportamento di questa classe ma sarebbero inutili in quanto questo controllo viene in realtà fatto dal parser.

Substitution cipher

Anche il substitution cipher è un cifrario abbastanza semplice che utilizza come chiave di cifratura una permutazione dell'alfabeto. Se per esempio la permutazione utilizzata è *qwertyuiopasdfghjklzxcvbnm* la lettera *a* verrà sostituita con la *q*, la *b* con la *w* e così via fino a scambiare la *z* con la *m*.

I comportamenti importanti da testare per questo cifrario sono ovviamente la cifratura e la decifratura ma anche il corretto settaggio della permutazione dell'alfabeto da utilizzare come chiave.

Cifratura e decifratura vengono testate con stringhe lunghe zero (stringa vuota), uno e quattro caratteri (questa scelta è dovuta solamente per cercare la cifratura di una stringa di senso compiuto, in questo caso *test*). La permutazione utilizzata è appunto quella specificata nell'esempio iniziale.

Come prima, i test riguardanti il controllo della permutazione di cifratura in realtà non sarebbero necessari in quanto già presenti nei test del parser (che è quello che si occupa di questo controllo) ma sono stati aggiunti per specificare meglio il comportamento che deve avere la classe in caso di inserimento di una permutazione scorretta (troppo corta, troppo lunga o con caratteri non alfabetici).

Mocking

MUTATION TESTING

INTEGRATION TESTING

TEST SULL'INTERFACCIA

3

PLUG-IN UTILIZZATI

MAVEN

TRAVIS

COVERALLS

SONARQUBE

4

CONCLUSIONI

A

SVOLGIMENTO COMPLETO ESERCIZIO 5.4

$$
\begin{aligned}
\mathbf{SSSSSSS} &= (\lambda xyz.xz(yz))\mathbf{SSSSSS} \\
&= (\lambda yz.\mathbf{S}z(yz))\mathbf{SSSSS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{S}z))\mathbf{SSSS} \\
&= \mathbf{SS}(\mathbf{SS})\mathbf{SSS} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SS})\mathbf{SSS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SS})z)\mathbf{SSS} \\
&= \mathbf{SS}(\mathbf{SSS})\mathbf{SS} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSS})\mathbf{SS} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SSS}z))\mathbf{SS} \\
&= \mathbf{SS}(\mathbf{SSSS})\mathbf{S} \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSSS})\mathbf{S} \\
&= (\lambda z.\mathbf{S}z(\mathbf{SSSS}z))\mathbf{S} \\
&= \mathbf{SS}(\mathbf{SSSSS}) \\
&= (\lambda yz.\mathbf{S}z(yz))(\mathbf{SSSSS}) \\
&= \lambda z.\mathbf{S}z(\mathbf{SSSSS}z) \\
&= \lambda z.(\lambda ya.a(ya))(\mathbf{SSSSS}z) \\
&= \lambda za.aa(\mathbf{SSSSS}aa) \\
&= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))\mathbf{SSS}zz) \\
&= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{S}b))\mathbf{SS}zz) \\
&= \lambda za.aa(\mathbf{SS}(\mathbf{SS})\mathbf{S}aa) \\
&= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))\mathbf{SSS}zz) \\
&= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{S}b))\mathbf{SS}zz) \\
&= \lambda za.aa(\mathbf{SS}(\mathbf{SS})\mathbf{S}aa)
\end{aligned}
$$

$$= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))(\mathbf{SS})\mathbf{S}zz)$$
$$= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{SS}b))\mathbf{S}zz)$$
$$= \lambda za.aa(\mathbf{SS}(\mathbf{SSS})aa)$$
$$= \lambda za.aa((\lambda yb.\mathbf{S}b(yb))(\mathbf{SSS})zz)$$
$$= \lambda za.aa((\lambda b.\mathbf{S}b(\mathbf{SSS}b))zz)$$
$$= \lambda za.aa(\mathbf{S}a(\mathbf{SSS}a)a)$$
$$= \lambda za.aa((\lambda yb.bb(yb))(\mathbf{SSS}z)$$
$$= \lambda za.aa((\lambda b.bb(\mathbf{SSS}bb))z)$$
$$= \lambda za.aa(aa(\mathbf{SSS}a))$$
$$= \lambda za.aa(aa((\lambda yb.\mathbf{S}b(yb))\mathbf{S}zz))$$
$$= \lambda za.aa(aa(\mathbf{S}a(\mathbf{S}a)a))$$
$$= \lambda za.aa(a((\lambda yb.bb(yb))(\mathbf{S}z)z))$$
$$= \lambda za.aa(aa((\lambda b.bb(\mathbf{S}bb))z))$$
$$= \lambda za.aa(aa(aa(\mathbf{S}aa)))$$
$$= \lambda za.aa(aa(aa((\lambda yb.bb(yb))z)))$$
$$= \lambda za.aa(aa(aa(\lambda b.bb(bb))))$$