Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Домашна задача бр.2
Криптографија

Проточен шифрувач – RC4

Стефан Андонов, 151020
stefan.andonov@students.finki.ukim.mk

Во оваа домашна задача ќе прикажам моја сопствена имплементација на проточниот шифрувач RC4 со помош на програмскиот јазик Java. Целосната имплементација е дадена со класта RC4ChiperTester.java. Шифрувачот е прикажан со класата RC4 којашто содржи неколку функции:

```
public RC4 ()
```

Конструктор за класата. Класата чува низата од 256 цели броеви, која што ја претставува т.н. внатрешна состојба на шифрувачот.

```
public void KSA (int [] key)
```

Оваа функцијата го симулира Key-scheduling алгоритамот на шифрувачот. Како аргумент прима низа од цели броеви, што го претставува хексадекадниот запис на клучот, но прикажан бајт по бајт. Пример клучот 0x0102030405 се преставува како низата {0x01,0x02,0x03,0x04,0x05}.

Најпрво, функцијата ја иницира низата S, така што секој член си ја добива вредноста на својот реден број.

Потоа, со помош на променливата j, се врши распоредувањето на клучот во низата S.

```
public List<Integer> PRGA (int [] key, int offset)
```

Оваа функција го симулира Pseudo-random generation algorithm, како и генерирањето на протокот на клучеви со должина 16 (како што се во тест-векторите). Прима два аргументи, низа од цели броеви која го претставува клучот, и поместувањето. Како резултат дава листа од цели броеви, што го претставува протокот на клучевите, секој со должина од 16 бајта.

На почеток вршиме распоредување на клучот (првиот аргумент) со помош на функцијата KSA(int [] key). Потоа го применуваме PRGA алгоритмот како што е дефиниран во предавањата и аудиториските вежби, и ги ставаме сите бајти чијшто реден број на добивање е поголем од дадениот offset во една листа, која што на крај ја враќаме како резултат од функцијата.

```
public String encryption (int [] key, String original)
```

Функцијата прави енкрипција на дадена порака. На влез прима два аргументи, клучот претставен како низа од цели броеви и оригиналната порака, како стринг.

Прво го добива соодветниот проток на клучеви со повикување на функцијата PRGA(). За offset е избрано да биде, должината на пораката – вториот аргумент на функцијата. Во функцијата декларираме една листа од цели броеви во која што ќе ги чуваме бајтовите кои што ќе ги добиваме еден по еден со примена на операцијата XOR помеѓу секој бајт од клучот (елемент од низата key) и секој бајт од пораката (секој карактер во стингот). На крај, со

помош на StringBuilder го основаме зборот којшто треба да го вратиме како резултат од функцијата.

```
public String decryption (int [] key, String encrypted)
```

Оваа функција во главно е наполно идентична како функцијата за енкрипција. Повторно се темели на XOR операцијата.

Во главната класа RC4ChiperTester правиме тестирање на тест-векторите, а резултатите може да се погледнат во датотеката output.txt. Тие се идентични како и на сајтот со тест-векторите.

Дополнително, имаме една порака која што ја шифрираме со повик на функцијата RC4::encryption(), а резултатот од функцијата го дешифрираме со повик на функцијата RC4::decryption(). Ја добиваме истата порака која што сме ја претставиле на почетокот.

Со тоа ја покажуваме успешноста на имплементацијата на шифрувачот RC4.