Домашна задача бр.4
Криптографија

ECB и CBC модови на работа. „Исечи-па-залепи“ напад

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

Во оваа домашна задача, ќе се прикаже работата на ECB (Electronic Code Book) мод на работа, како и на CBC (Cipher block chaining) мод на работа на еден блоковски шифрувач, но и ќе го симулирам нападот врз шифриран текст – „исечи-па-залепи“.

За потребите на домашната задача најпрво дефинирам посебен блок шифрувач, чија што програмска имплементација е дадена во код во Јава, достапен на GitHub.

Блоковски шифрувач и модули за ECB и CBC енкрипција и декрипција

Блоковскиот шифрувач работи со блокови од 64 битови (8 ASCII карактери, секој по 8 битови). Клучот е целобројна вредност – integer, со максимум 32 бита. Шифрирањето се заснова на Цезаровиот шифрувач, односно shifting cipher. Дополнително, за разлика од обичниот шифрувач, тука е дозволено во пораката да има бројки и специјални знаци, при што во процесот на шифрирање, се врши поместување за одреден број на позиции според клучот и на нив.

Еден блок е претставен со класата Block.java, која што содржи променлива од типот String – block, и истиот ги имплементира следните функции:

- `public String encrypt (int key)`
  Функција која врши шифрирање во нормален мод (ECB). На секој карактер во низата од карактери block, доколку е буква му го додава клучот по модул 26, доколку е специјален карактер му го додава клучот по модул 16, а доколку е бројка по модул 10
- `public String encryptCBC (int key, long iv)`
  Функција која врши енкрипција според модот на работа CBC. Пред да ја изврши претходната дефинираната функција encrypt(int key), прави XOR операција на block и на иницијалниот вектор iv.
- `public String decrypt(int key)`
  Функција која врши декрипција на блокот во модот ECB. Декрипцијата е инверзна од енкрипцијата, односно сега го одземаме клучот по соодветен модул
- `public String decryptCBC(int key, long iv)`
  Фрши декрипција на блокот според модот CBC. Истата се врши на следниот начин:

$$x_1 = D_k(y_1)\ xor\ IV$$
$$x_i = D_k(y_i)\ xor\ y_{i-1}$$

За овие функции да функционираат дополнително се дефинирани и помошните функции, ceaserEncryption и ceaserDecryption, како и се користат функциите направени во претходната домашна задача: Long StrToLong(String s) и String LongToStr(Long l).

```
public String ceasarEncryption (String m, int key){
		StringBuilder sb = new StringBuilder();

		for (int i=0;i<m.length();i++){
			char c = m.charAt(i);
			if ('a'<=c && c<='z'){
				sb.append((char)((c-'a'+key)%26+'a'));
			}
			else if ('A'<=c && c<='Z'){
				sb.append((char)((c-'A'+ key)%26+'A'));
			}
			else if ('0' <= c && c<='9'){
				sb.append((char) ((c-'0'+key)%10+'0'));
			}
			else if (' '<=c && c<='/'){
				sb.append((char) ((c-' '+key)%16+' '));
			}
			else
				sb.append(c);
		}
		return sb.toString();
	}

	public String ceasarDecryption(String m, int key){
		StringBuilder sb = new StringBuilder();
		for (int i=0;i<m.length();i++){
			char c = m.charAt(i);
			if ('a'<=c && c<='z'){
				char c1 = (char)((c-'a'-key)%26+'a');
				if (c1<'a')
					c1=(char) (c1+'z'-'a'+1);
				sb.append(c1);
			}
			else if ('A'<=c && c<='Z'){
				char c1 = (char)((c-'A'-key)%26+'A');
				if (c1<'A')
					c1=(char) (c1+'Z'-'A'+1);
				sb.append(c1);
			}
			else if ('0' <= c && c<='9'){
				char c1 = (char)((c-'0'-key)%10+'0');
				if (c1<'0')
					c1=(char) (c1+'9'-'0'+1);
				sb.append(c1);
			}
			else if (' '<=c && c<='/'){
				char c1 = (char)((c-' '-key)%16+' ');
				if (c1<' ')
					c1=(char) (c1+'/'-' '+1);
				sb.append(c1);
			}
			else
				sb.append(c);
		}
		return sb.toString();
	}
```

```
public String encrypt (int key){
	return ceasarEncryption(block,key);
}

public String encryptCBC (int key, long iv){
	long input = iv ^ Block.StrToLong(block);
	return ceasarEncryption(LongToStr(input),key);
}

public String decrypt(int key) {
		return ceasarDecryption(block,key);
}

public String decryptCBC(int key, long iv){
	long ret = iv ^ StrToLong(ceasarDecryption(block,key));
	return LongToStr(ret);
}
```

Имплементација на „Исечи-па-залепи“ напад

Овој напад се извршува врз шифриран текст и може да биде извршен на 2 начини:
- бришење на еден или повеќе блокови од шифрираниот текст
- замена на местата на некои блокови во шифрираниот текст

Целта на нападот е кога примачот на пораката ќе ја добие шифрираната (сечена, лепена и монтирана) порака, да не забелеши дека нешто е променето во пораката и истата да ја перцепира како точна. Тоа е имплементирано со следните функции:

```
public String cutBlock(int numBlock) {
	return blocks.get(numBlock-1);
}

public void swapBlocks(int b1, int b2) {
	List<String> newBlocks = new ArrayList<>();
	int pom;
	if (b1>b2){
		pom=b2;
		b2=b1;
		b1=pom;
	}
	String block1 = cutBlock(b1);
	String block2 = cutBlock(b2);
	newBlocks.addAll(blocks.subList(0, b1-1));
	newBlocks.add(block2);
	newBlocks.addAll(blocks.subList(b1, b2-1));
	newBlocks.add(block1);
	newBlocks.addAll(blocks.subList(b2, blocks.size()));

	blocks=newBlocks;
}

public void deleteBlock(int bNumber){
	blocks.remove(bNumber-1);
}
```

Поконкретно ќе може да се забележи на тест примерите подолу:

Тест пример 1:
Message: Money_for_Alice_is_$100 Money_for_Trudy_is_$2___
Message into blocks: Money_fo|r_Alice_|is_$100_|Money_fo|r_Trudy_|is_$2___|

Пораката е поделена во точно 6 блокови. Доколку ги остраниме 3иот и 6тиот блок и потоа им ги смениме местата би требало да ја добиеме следната порака:
Money_fo|r_Alice_| is_$2___| Money_fo|r_Trudy_| is_$100_|

Да видиме дали тоа функционира со двата мода на работа:
ECB мод:
```
String message = "Money for Alice is $100 Money for Trudy is $2";
BlockChiper bc = new BlockChiper(message,0);
System.out.println(bc.toString());
String c = bc.encrypt(10);
System.out.println("Enkriptiranata poraka e: "+c);
BlockChiper bc1 = new BlockChiper(c,0);
bc1.swapBlocks(3, 6);
System.out.println(bc1.toString());
String m = bc1.decrypt(10);
System.out.println("Dekriptirata poraka posle napad cut-and-paste: "+m);
```

```
[Money fo, r Alice , is $100 , Money fo, r Trudy , is $2   ]
Enkriptiranata poraka e: Wyxoi*pyb*Kvsmo*sc*.100*Wyxoi*pyb*Dbeni*sc*.2***
[Wyxoi*py, b*Kvsmo*, sc*.2***, Wyxoi*py, b*Dbeni*, sc*.100*]
Dekriptirata poraka posle napad cut-and-paste: Money for Alice is $2   Money for
Trudy is $100
```

CBC мод:
```
String message = "Money for Alice is $100 Money for Trudy is $2";
BlockChiper bc = new BlockChiper(message,0xabcfff1211045fffL);
System.out.println(bc.toString());
String c = bc.encryptCBC(10);
System.out.println("Enkriptiranata poraka e: "+c);
BlockChiper bc1 = new BlockChiper(c,0xabcfff1211045fffL);
bc1.swapBlocks(3, 6);
System.out.println(bc1.toString());
String m = bc1.decryptCBC(10);
System.out.println("Dekriptirata poraka posle napad cut-and-paste: "+m);
```

```
[Money fo, r Alice , is $100 , Money fo, r Trudy , is $2   ]
Enkriptiranata poraka e: ? ?gr.9????__W\°???)$qv????V]A_????."/s????
__C?
[? ?gr.9?, ???__W\°, ???
__C?, ???V]A_?, ???."/s?, ???)$qv?]
Dekriptirata poraka posle napad cut-and-paste: Money for Alice ?O:__X_O_StFM^S_
```

Можеме да заклучиме дека Cut-and-paste нападот е успешен на оваа порака доколку имаме ECB енкрипција и декрипција, но доколку имаме CBC не е успешен, бидејќи доаѓа до промени во пропагацијата на иницијалниот вектор, па со тоа немаме успешна декрипција.

Доколку сакаме да извршиме напад со бришење на блок, пример на истиот тест пример, сакаме да ги избришеме блоковите 2,3 и 4, со што ќе креираме илузија дека оригиналната порака е Money for Trudy is $2___.

ECB Мод:

```
String message = "Money for Alice is $100 Money for Trudy is $2";
BlockChiper bc = new BlockChiper(message,0xabcfff1211045fffL);
System.out.println(bc.toString());
String c = bc.encrypt(10);
System.out.println("Enkriptiranata poraka e: "+c);
BlockChiper bc1 = new BlockChiper(c,0xabcfff1211045fffL);
bc1.deleteBlock(2);
bc1.deleteBlock(2);
bc1.deleteBlock(2);
System.out.println(bc1.toString());
String m = bc1.decrypt(10);
System.out.println("Dekriptirata poraka posle napad cut-and-paste: "+m);
```

```
[Money fo, r Alice , is $100 , Money fo, r Trudy , is $2   ]
Enkriptiranata poraka e: Wyxoi*pyb*Kvsmo*sc*.100*Wyxoi*pyb*Dbeni*sc*.2***
[Wyxoi*py, b*Dbeni*, sc*.2***]
Dekriptirata poraka posle napad cut-and-paste: Money for Trudy is $2
```

CBC Мод:

```
String message = "Money for Alice is $100 Money for Trudy is $2";
BlockChiper bc = new BlockChiper(message,0xabcfff1211045fffL);
System.out.println(bc.toString());
String c = bc.encryptCBC(10);
System.out.println("Enkriptiranata poraka e: "+c);
BlockChiper bc1 = new BlockChiper(c,0xabcfff1211045fffL);
bc1.deleteBlock(2);
bc1.deleteBlock(2);
bc1.deleteBlock(2);
System.out.println(bc1.toString());
String m = bc1.decryptCBC(10);
System.out.println("Dekriptirata poraka posle napad cut-and-paste: "+m);
```

```
[Money fo, r Alice , is $100 , Money fo, r Trudy , is $2   ]
Enkriptiranata poraka e: ? ?gr.9????__W\°???)$qv????V]A_????."/s????
__C?
[? ?gr.9?, ???."/s?, ???
__C?]
Dekriptirata poraka posle napad cut-and-paste: Money fo$_[CZ_POis $2
```

Нападот повторно, како и претходно, е успешен при ECB мод на работа, но при CBC мод на работа истиот е неуспешен, поради промените во пропагацијата на иницијалниот вектор со бришењето на трите блока.