Информациска Безбедност Лаб 01

При имплементација на првата лабораториска вежба решив да тргнам прво со наоѓање на податоци кои може да се искористат како Ип Пакети, соодветно на тоа побарав на интернет за веќе достапни снимани пакети за 802.11 протоколот.
Документот со пакети се наоѓа во utils/packets.csv

Код за претварање од редови во датетоката во Објекти од ИпПакети:

```
public static List<Frame> ofFile() {
    try {
        BufferedReader bufferedReader = new BufferedReader(new
FileReader("src/lab_01/utils/packets.csv"));
        return bufferedReader.lines().skip(1)
                .map(ClearTextFrame::of)
                .collect(Collectors.toList());
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    return new ArrayList<>();
}
```

Архитектурата на **Frame** е поделена во една апстрактна класа и две под класи од Тип на **ClearTextFrame** I **EncryptedFrame.**

```
public abstract class Frame {
    private FrameHeader header;
    private Integer packageNumber;
    private FrameData data;


    public Frame(FrameHeader header, Integer packageNumber, FrameData data) {
        this.header = header;
        this.packageNumber = packageNumber;
        this.data = data;

    }

    public abstract MIC calculateMic(String secret, byte[] nonce, MIC mic);

    public FrameHeader getHeader() {
        return header;
    }

    public Integer getPackageNumber() {
        return packageNumber;
    }

    public EncryptedFrame encrypt(String secret, byte[] nonce) {
        byte[] counter = new byte[]{0, 0, 0};
        CTRPreload preload = new CTRPreload(nonce, counter);
        List<byte[]> dataBlocks = this.getData().getByteData();
        List<byte[]> chainEncryptedData = new ArrayList<>();
        byte[] chain = AES.encrypt(preload.toBytes(true), secret);
```

```
        byte[] xor = Utils.xor(chain, dataBlocks.get(0));
        chainEncryptedData.add(xor);
        for (int i = 1; i < dataBlocks.size(); i++) {
            chain = AES.encrypt(preload.toBytes(true), secret);
            xor = Utils.xor(dataBlocks.get(i), chain);
            chainEncryptedData.add(xor);
        }

        return new EncryptedFrame(this.header, this.packageNumber, new
FrameData(chainEncryptedData));
    }

    public ClearTextFrame decrypt(String secret, byte[] nonce) {
        byte[] counter = new byte[]{0, 0, 0};
        CTRPreload preload = new CTRPreload(nonce, counter);
        List<byte[]> dataBlocks = this.getData().getByteData();
        byte[] chain = AES.encrypt(preload.toBytes(true), secret);
        byte[] xor = Utils.xor(dataBlocks.get(0), chain);
        List<byte[]> clearTextFrameBlocks = new ArrayList<>();
        clearTextFrameBlocks.add(xor);
        for (int i = 1; i < dataBlocks.size(); i++) {
            chain = AES.encrypt(preload.toBytes(true), secret);
            clearTextFrameBlocks.add(Utils.xor(dataBlocks.get(i), chain));
        }
        return new ClearTextFrame(getHeader(), getPackageNumber(), new
FrameData(clearTextFrameBlocks));
    }

    public FrameData getData() {
        return data;
    }

    public abstract MIC getMic();

    public byte[] getPackageNumberBytes() {
        return new byte[]{packageNumber.byteValue()};
    }

    public static Integer packageNumberOf(String line) {
        String[] parts = line.replace("\"", "").split(",");
        return Integer.parseInt(parts[0]);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Frame frame = (Frame) o;
        return this.getData().getByteData().equals(frame.getData().getByteData()) &&
                this.getHeader().equals(frame.header) &&
                this.packageNumber.equals(frame.getPackageNumber());
    }

    @Override
    public int hashCode() {
        return Objects.hash(data);
```

```
    }

    public void print() {

        StringBuilder es = new StringBuilder();
        for (byte[] array : this.getData().getByteData()) {
            for (byte b : array) {
                es.append(String.format("%02X ", b));
            }
        }
        byte[] newStringByte = new byte[this.getData().getByteData().size() * 16];
        int i = 0;
        for (byte[] array : this.getData().getByteData()) {
            System.arraycopy(array, 0, newStringByte, i * array.length, array.length);
            i++;
        }
        System.out.println(es.toString());
        System.out.println(new String(newStringByte));


    }

}
```

Другите две подкласи имаат само дополнително MIC .

Во делот со CCMP

```
public class CCMP {
    private List<Frame> clearTextFrame;


    public CCMP(List<Frame> clearTextFrame) {
        this.clearTextFrame = clearTextFrame;
    }


    private byte[] calculateNonce(Frame frame) {
        byte[] nonce = new byte[13];
        System.arraycopy(frame.getPackageNumberBytes(), 0, nonce, 0,
frame.getPackageNumberBytes().length);
        if (frame.getHeader().getSourceIpAddressBytes().length > 12) {
            System.arraycopy(frame.getHeader().getSourceIpAddressBytes(), 0, nonce,
frame.getPackageNumberBytes().length,
                    12);
        }
        return nonce;
    }

    public boolean match(String secret) {
        for (int i = 0; i < clearTextFrame.size(); i++) {
            Frame frame = this.clearTextFrame.get(i);
            Frame frameEncrypted = frame.encrypt(secret, calculateNonce(frame));
            frame.calculateMic(secret, calculateNonce(frame), frame.getMic());
            frameEncrypted.calculateMic(secret, calculateNonce(frame), frame.getMic());
            Frame toTest = frameEncrypted.decrypt(secret, calculateNonce(frame));
```

```
            MIC mic1 = frame.calculateMic(secret, calculateNonce(frame), null);
            MIC mic2 = toTest.calculateMic(secret, calculateNonce(toTest), null);
            assert (verify(mic1, mic2));
            System.out.println("Authentication successful");
            frame.print();
            assert (frame.equals(toTest));
            System.out.println("Decryption successful");
            toTest.print();
        }
        return true;

    }

    public boolean verify(MIC mic1, MIC mic2) {
        assert (mic1.equals(mic2));
        return true;
    }


}
```

Примам листа од пакети со чист текст ги криптирам и верификувам дали добиениот мик е ист .
До делот за пресметување на нонсе не внесувам Qos бидејќи го немам во делот со пакетите , па се служам со број на пакет и мак адреса.

Изработил:
Ќазим Демиров 153081
Примена на е-технологии