**Cyber Security Assessed Exercise**

**Report**

**Anton Belev 1103816b**

1. **Known Plain Text Attack**

***Message was***: “Do what's right and try to get along with people, in that order”

***Key was:*** “0xb4db”

*KPT.java* contains a main method. When this java file is executed, it will perform a brute force attack to find the key that encrypted the message. After the end of the execution the found key and the decrypted messages are printed to the standard I/O.

1. **Cipher Text Only Attack**

***Message was***: “Do, or do not. There is no try”

***Key was:*** “0xfd1f”

*CTO.java* contains a main method. When this file is executed it will print out the key and the message. Here I’m using an English dictionary from words.txt file and bigram frequency statistics from bigrams.txt. These files help me to guess how likely the decrypted message is valid English. I’m using a scoring system. After the decryption I split the text by space and for each word I’m checking if the word is in the dictionary, if not I take away from the score. Here the bigger the word the higher the weight. Also bigrams.txt is penalizing bigrams that are not likely English. Both words.txt and bigrams.txt are not my own work and they are found on the Internet.

**Experiment**:

Just after I found the most likely key, I’m performing an experiment to see how many cipherblocks I need to correctly decrypt the message. I’m testing for each possible number of blocks (in my case 1,2...16), and for each of them, I’m taking each possible consecutive sequence of blocks with that length. The result of the experiment is printed as well. **This may take 3-4mins to finish.**

Start of experiment...

Results...

Ciphertext blocks used: 1. In total 0/16 attacks have found the correct key.

Ciphertext blocks used: 2. In total 0/15 attacks have found the correct key.

Ciphertext blocks used: 3. In total 2/14 attacks have found the correct key.

Ciphertext blocks used: 4. In total 4/13 attacks have found the correct key.

Ciphertext blocks used: 5. In total 10/12 attacks have found the correct key.

Ciphertext blocks used: 6. In total 8/11 attacks have found the correct key.

Ciphertext blocks used: 7. In total 8/10 attacks have found the correct key.

Ciphertext blocks used: 8. In total 9/9 attacks have found the correct key.

Ciphertext blocks used: 9. In total 8/8 attacks have found the correct key.

Ciphertext blocks used: 10. In total 7/7 attacks have found the correct key.

Ciphertext blocks used: 11. In total 6/6 attacks have found the correct key.

Ciphertext blocks used: 12. In total 5/5 attacks have found the correct key.

Ciphertext blocks used: 13. In total 4/4 attacks have found the correct key.

Ciphertext blocks used: 14. In total 3/3 attacks have found the correct key.

Ciphertext blocks used: 15. In total 2/2 attacks have found the correct key.

Ciphertext blocks used: 16. In total 1/1 attacks have found the correct key.

**Theoretical Calculation:**

The key is 16 bit. *H(K)* = 16

Brute force attack needs to try 216 keys. **Absolute rate** *R* for English language (assuming all sequence of characters are **equally likely**) *R* = log2(26) = 4.7

**Actual rate** *r* is between 1.0 and 1.5. Take *r* = 1.5

**Redundancy** *D* = *R – r* = 3.2

**Utility distance** *H(K)/D = 16/3.2 = 5*

This means that we need 5 decrypted letters to make sure a massage that looks like English is the real message.

In our case 1 block is 2 letters. So we need approximately 3 cipherblocks.

As you can see from the experiment I was able to successfully find the key using only 3 cipherblocks, but it really depends which one you pick.

1. **Time Memory Tradeoff**

***Message was***: “Victory goes to the player who makes the next-to-last mistake”

***Key was:*** “0x4564”

*TMT1.java* contains a main method. When this java file is executed, it will create table.txt file containing the time memory tradeoff table.

*TMT2.java* contains a main method. When this java file is executed, it will use the table created from TMT1.java to find the key and decrypt the message.

1. **Other Comments**

Additionally to the report, KTM, CTO, TMT1 and TMT2, I have attached words.txt, bigrams.txt and Utils.java. Utils.java is an abstract class containing methods that are extensively used in all the java files.