Cyber Security - Assessed Exercise Report

Michael Kilian - 1003819k

February 21, 2014

1 Known Plain Text Brute Force

The KPT program tries all possible keys, and stops when a key is found for which the first cipher text block is correctly decrypted to the known plaintext. It returns this key once found. Feeding this in to the DecryptAllBlocks program retrieves the plaintext. This was then read using the Block2Text program provided.

The key found was 0x96B2. The message was *"Life is too short to slay each dragon more than once"*.

2 Cipher Text Only Attack

2.1 Conducting the attack

The CTO program makes use of a monogram frequency table to recognise a valid message. This is read in from a file[1] (in the frequency-data folder) and built as a map. The table in the file does not have frequencies associated with n-grams; instead it simply lists the possible n-grams in order of frequency, most frequent to least frequent. For example, the monograms file lists all possible English characters (including punctuation, newlines and numbers) starting with the space character and continuing in decreasing order of frequency.

To assign a numeric value to each n-gram, the following scheme was adopted: let N be the number of n-grams in the set. The first n-gram is assigned N, the second is assigned N-1, etc. Lookup of n-grams is case insensitive; no assumptions are made regarding the placement of capital letters in the sentence. This is largely because accurate frequency tables which included capital letters could not be found.

Suppose we are considering n-grams of order n. Given a piece of text, we assign a score to it with the following procedure:

Data: String text
Result: Integer score
Total_Score = 0;
for *Each n-gram n in the document* **do**
  s = look up n in frequency table;
  if *s != 0* **then**
    Total_Score + = $s * 10^n$;
  end
end
return Total_Score

Algorithm 1: N-gram scoring procedure.

The total score for a document is the sum of its score for each n-gram category we are considering (in this case, monograms). The scaling by powers of ten ensures that matches to n-grams of larger order are given more weight. So for example, we assume that a match to a bigram is more indicative of correct english than a match to a monogram. Methods are provided for building frequency tables and scoring text for n-grams of arbitrary order.

Assuming we have built the frequency tables correctly, the following procedure is used to attack the key space:

Data:
Result: Integer score
String c = cipher text blocks;
Double bestScore = 0;
Integer bestKey = -1;
for *Each possible key k* **do**
  Double Score = 0;
  p = decrypt(k,c);
  if *p contains non-english characters* **then**
    continue;
  end
  for *Each order n of ngram to be considered* **do**
    score + = ngramScore(p,n);
  end
  if *score > bestScore* **then**
    bestScore = score;
    bestKey = k;
  end
end
return bestKey;

Algorithm 2: Pseudocode of CTO attack.

In other words, for every key we decrypt the text and score the result. We then return the key which resulted in the highest scoring text. Any messages which contained any non-valid english characters can be thrown away immediately, where valid english characters are defined by the set of monograms in the monogram frequency table.

2.2 Theoretical Calcualtion of Unicity Distance

Each character is stored using 1 byte (8 bits). Therefore the absolute rate of the language is $R = 8$. We also know that the actual rate for english can be

defined as $r = 1.5$. Using the formula:

$$D = R - r \tag{1}$$

we can calculate the redundancy D of the language to be 6.5.

We now consider the entropy of the key space. The following relationship gives the entropy H(K) of the key space :

$$H(K) = \sum p(k_i) * log(1/k_i) \tag{2}$$

The key space is the set of 16 bit numbers. Assuming all keys are equally likely, using the above formula we conclude the entropy of the key space is $H(K) = 16$. This result trivially follows from the definition of entropy as "the number of bits required to encode all possible messages in a language with an optimal encoding".

Finally, we calculate the Unicity Distance (the minimum number of characters we must correctly decrypt in order to know we have unambiguously decoded the message) N_u. This is defined as:

$$N_u = H(K)/D \tag{3}$$

Plugging in the results of Equations 1 and 2, we conclude that the unicity distance is 2.461. Therefore we must decrypt roughly 3 characters to know we have decoded the message correctly. Since each block contains two 1 byte characters, we conclude that we require at least 2 blocks to be decrypted correctly to break the code unambiguously.

2.3 Experiment and Results

The CTO program takes a number of blocks to read as one of its parameters. The message was first decrypted using the maximum block size, giving the correct message for comparison and the key used to find it.

To find the actual number of blocks needed to break the message, the program was ran repeatedly reducing the number of block analysed on each run. The actual minimum number of blocks required to select the correct message was 5, significantly higher than the theoretical calculation of 2.

The message found was *"Keep your lies consistent - Ferengi Rule of Acquisition #60\n"*. It was found with key 0x5BA2.

2.4 Design Choices

Originally the CTO attack was tried with bigrams also; this was removed however as it actually increased the number of blocks needed to decode the message. This is probably a result of an incomplete or inaccurate bigram frequency table, or the result of the scoring system used to rate n-grams. In particular, the bigram table used (taken from the same source as the monogram table) did not include pairings of letters and spaces, which are naturally quite common. The result was that messages which were valid english but did not contain many spaces were given a significantly higher rating than the actual message. As mentioned the methods provided in the program can handle arbitrary n-gram lengths, so with more accurate frequency tables it would be possible to reduce the number of blocks needed.

3 Time Memory Tradeoff

To correctly construct the table, we must choose appropriate values for N (the number of chains) and L(the length of each chain). Let T be the number of possible keys. With 16 bits keys, T = 2^{16}. In order to ensure our table has the right number of keys, we must select L,N such that:

$$T = L * N \tag{4}$$

L = 2^8 and N = 2^8 were chosen. Clearly Equation 4 holds for the aforementioned values of L,N and T. Though this guarantees the right number of keys, the table will almost certainly contain duplicates and so it does not guarantee that all possible keys are in the table.

Program TMT1 builds the table and saves it to a file, as (x_L,x_0) pairs, where x_0,x_L are the first and last values of the chain respectively. It calculates each chain by starting with a random value in the range [0,2^{16}]. It then performs L encryptions to obtain the end of the chain, as described in Lecture 7. N such chains are created, each using its own random value. After each chain is computed it is written to the table file. The first block of plaintext is encoded into the program for use in the chain computation process.

Program TMT2 reads in the output file from TMT1 and constructs a TMT table using the provided Table class. It also reads in the first block of ciphertext and contains the first block of plaintext. It then performs the TMT lookup attack using the cipher and plaintext block. It stops when a key is found which correctly decrypts the first ciphertext block to the plaintext block, or after searching the table if the key is not contained in the table. If found, the program then prints out the key to standard output. As for the KPT attack, the provided Block2Text and DecryptAllBlocks programs were used to decrypt and read the cipher message using this key.

The message found was *"Reality is that which, when you stop believing in it, doesn't go awa"*. I am presuming that part of the message provided in the email was actually missing as this does not form a complete sentence, and the message was decrypted using the provided programs. I have triple checked that the ciphertext provided in the email leads to the above. The key found was 0x209E.

4 Running the programs

The programs should be compiled and run in the default folder along with the provided programs/classes on which they depend. The *data* folder contains the cipher/plaintext message provided, where the exercises KPT,CTO and TMT have been numbered 1,2 and 3 respectively. Each program provides a usage statement explaining the required command line input needed.

In particular, the CTO program relies on a monogram frequency table, included in the *frequency-data* folder. This folder MUST be included in the same folder as the CTO program when running it.

The programs are written in Java 6. They can be compiled using Javac in the usual manner.

References

[1] Source of frequency tables, *http://mdickens.me/typing/letter_frequency.html*. Note: Accessed 20/02/14