Worksheet 1 (due by Monday June 13, 11:59 pm)

Course 01435: Practical Cryptanalysis
June 2016

Andrey Bogdanov
anbog@dtu.dk

Introductory Remarks

All comments from worksheet 0 apply. In particular, it is recommended to concentrate on portfolio exercises and the programming project if time is short.

TMTO 1 (Hellman Tables)

Recommended Reading

The original paper by M. Hellman (*A Cryptanalytic Time-Memory Trade-Off*) is available from CampusNet. For those preferring an easier read, Chapter 4.4 in the book *Applied Cryptanalysis: Breaking Ciphers in the Real World* by M. Stamp and R. Low gives an introduction to the topic and is available online for those with a DTU login:

http://lib.myilibrary.com.globalproxy.cvt.dk/Browse/Open.asp?ID=90120

Exercises

Exercise 6: Until the year 2000, U.S. companies were not allowed to export cryptographic algorithms with key lengths of more than 40 bits. If the key was longer (as e.g. for DES), the remaining bits had to be published (e.g. set to zero).

Given your findings from exercise 5, how long would it take you to compute a full table of all possible 40-bit keys? What would be the size of that table, and how would you store it?

Exercise 7: Modify the program from exercise 5 such that your plaintext/key length is determined by a program parameter called BLOCKSIZE (e.g. start with BLOCKSIZE = 20 bit). Now we want to implement a table lookup attack.

To this end, modify the program such that it now takes a known plaintext as input and stores the corresponding ciphertext for all keys in RAM. Observe

the performance of the program when BLOCKSIZE (and thus the table size) is increased. Draw the graph mapping the number of stored keys to the running time (e.g. using Excel). What happens as BLOCKSIZE gets large? Do you have an explanation?

Exercise 8: Create your own random function $f : \{0, \ldots, 19\} \to \{0, \ldots, 19\}$.[1] Draw the functional graph for this function f as well as for the functions f_1 and f_2 defined by $f_i(x) = f(x) + i \bmod 20$. Do they show any similarities?

Exercise 9 (P): Find an implementation of the hash function[2] MD5 that works with your programming language and write a wrapper function `md5-redux` that restricts the input and output size of MD5 to 20 bit (e.g. by just throwing away all exceeding output bits).

Generate 2^{16} chains of length 2^8 by drawing random starting points and iterating f 256 times. Keep track of how many different points in $\{0,1\}^{20}$ you have covered after i calls to MD5. Make a graph over how the number of covered points develops over time. Do your observations from the graph match the preditions from the lecture?

TMTO 2 (Practical Issues)

Recommended Reading

The original paper introducing Rainbow tables (*Making a Faster Cryptanalytic Time-Memory Trade-Off* by P. Oechslin) is available on CampusNet.

Exercises

Exercise 10 (P): Consider AES-128. What are the complexities/resource consumption for a TMTO attack using Hellman tables and using rainbow tables. Find the optimal values (best trade-off) for m, t and ℓ. Give a table that compares pre-computation, memory, online computations and table lookups.

Exercise 11 (P): Let $f : \{0,1\}^{20} \to \{0,1\}^{20}$ be the reduced MD5 function as defined in exercise 14. We now define functions f_i by setting $f_i(x) = f(x) \oplus i$. The task is to modify the program from exercise 15 such that it generates 2^{16} chains of length 2^8 in a Rainbow table fashion, i.e. in each chain, we use f_i to compute the i-th entry. Again, make a graph of how the number of covered points develops over time. How does it compare to the graph from exercise 15?

[1] E.g. use an online resource like http://www.randomizer.org/form.htm, write a short program, use a D20 roleplaying dice if you have one, or just make up some random-looking values.

[2] MD5 is no longer considered a cryptographically secure hash function. But it will do for our purposes and has the advantage that it is widely available.