Worksheet 0 (due Sunday June 12, 11:59 pm)

Course 01435: Practical Cryptanalysis
June 2016

Andrey Bogdanov
anbog@dtu.dk

Introductory Remarks

In the course "Practical Cryptanalysis", there is a set of exercises for every topic. Some of the exercises are optional and some exercises you are asked to solve at home and hand in. The latter are marked with (P) and should be handed in under CampusNet by the due date given on the worksheet.

- **Recommended Reading:** This course does not have an exam, thus, there is no pensum list and no "required reading". But of course, you are welcome to read more about the issues discussed in this course if you are interested or there is something you do not understand from the lectures. However, as with most specialist courses, there is not one textbook that can be recommended. Instead, you will have to piece together the course material from various sources.
- **Exercises:** Exercises can be of very different character. Some are pen-and-paper exercises, while others require computer aid or even dedicated programming. Unless otherwise stated, you are free to choose the means and tools by which you solve the exercise.
- **Programming Project:**(see extra worksheet) There is one programming project per week on a seperate worksheet. The programming project is mandatory, it is the core part of your final hand-in. If you don't hand in all programming projects, you will not pass the course, so make sure you spend enough time on them! On the other hand, you are free to choose your programming platform and language.

Lecture 1: Brute Force Cryptanalysis

Recommended Reading

If you have difficulties understanding the notions from the first lecture, you can learn more by using the Internet (in particular Wikipedia). No dedicated book

or paper on cryptography should be necessary.

Note that there are different ways to define the unicity distance. We used the redundancy of the language, however e.g. also entropy could be used to calculate the unicity distance but introducing entropy is out of the scope of this course.

Exercises

Exercise 1 (P): Decrypt the following ciphertext that was encrypted with a Caesar cipher:

```
GJBFW JYMJN IJXTK RFWHM
```

Exercise 2: In the lecture, we saw that the English words `arena` and `river` can have the same Caesar encryption, given the right key. Find other examples of words (in English or Danish) of length at least 4 that can have the same Caesar encryption.

Exercise 3 (P): Describe at least one real-life example for both known plaintext attacks (KPA) and chosen plaintext attacks (CPA). Remember to give a reference for where you found this particular attack.

Exercise 4(P): The plaintext alphabet of the Advanced Encryption Standard (AES) consists of the set $\{0,1\}^{128}$. The key set is $\{0,1\}^{128}$ for AES-128, $\{0,1\}^{192}$ for AES-192, and $\{0,1\}^{256}$ for AES-256. For each of the three AES versions, compute the unicity distance if the plaintext is English text. What does your result mean?

Exercise 5: (P) Implement a small function `encrypt(key,plaintext)` that takes two 128-bit values as input and returns their exclusive or as output. Even though this function is a lot simpler than a real encryption function (and completely insecure in most scenarios), we want to use it to get a feeling for how efficient brute force cryptanalysis is[1]:

- Given a known plaintext and ciphertext (128 bit each), write a loop that simulates a brute-force cryptanalysis, i.e. tests as many keys as possible. How many keys can you try out within 1 hour?
- Extend this estimate: How many keys could you try out within 1 year? What if you had a bot net of 1 million computers available? What key length can you break in this way?

Note that the code (hopefully less than 1 page) should be part of your hand-in.

[1] If you feel up to the task, you are welcome to use a library implementation of the AES instead.