Technische Universität Braunschweig

IAS INSTITUTE FOR APPLICATION SECURITY

Exercise Sheet 1

Introduction to IT-Security

Submit your solutions via Gitlab
Deadline: *Wednesday, November 19th, 09:00 a.m. CET 2025*

Symmetric Cryptography Fundamentals

1. *Security goals.* Alice returns home and finds the door to her appartment open. Mallory broke in while she was away.
   (a) (1 point) What security goals could have been violated? Give examples.
   (b) (1 point) Name security mechanisms from different classes that Alice could deploy to protect her belongings.

2. *Simple combinatorics.* How large is the key space for:
   (a) (1 point) ROT13?
   (b) (1 point) the Vigenère cipher with a known key length of n?
   (c) (1 point) AES with a 256 bit key?
   (d) (1 point) a monoalphabetic substitution cipher with k letters?

3. (4 points) Alice uses a XOR cipher to send two encrypted messages to Bob. She sends two ciphertexts $c_0 = m_0 \oplus k$ and $c_1 = m_1 \oplus k$ encrypted with the same key k. Eve captures both ciphertexts. Eve knows that the plaintext m must be one of the messages m_0 and m_1. How can Eve recover the other plaintext message? Can Eve also recover the exact key k used by Alice? Justify your answer.

Mono- and Polyalphabetic Ciphers

In this part of the exercise you will implement tools to encrypt and decrypt messages as well as automatically break encryptions. For submitting your solutions use our `Gitea` instance on `https://itsec.ias.tu-bs.de`. Fork the repository `Exercise-01` and activate the unit tests for the forked repository on `https://ci.itsec.ias.tu-bs.de` just as you did for the first sheet.

4. (3 points) Using the `argparse` module implement a tool that encrypts and decrypts a message using a monoalphabetic substitution cipher. Every letter of the alphabet hence should be mapped to a different letter according to a substitution key. Save the file in the `src/mono/` directory.

   ```
   usage: mono.py [-h] (--encrypt KEY | --decrypt KEY) [--out OUTFILE] FILE
   ```

   (a) Encryption:
      i. (1 point) The plaintext should be read from `FILE`: All non-alphabetic characters should be skipped and the remaining ones converted to lowercase.
      ii. (1 point) This text is then encrypted using the specified key alphabet and written to the output file `OUTFILE` or, if not specified, to standard output (`stdout`). The key is simple a string containing each character of the alphabet `a-z` exactly once.¸

   (b) Decryption:
      i. (1 point) Read the ciphertext from `FILE` and decrypt it using the specified key alphabet and write it to standard output (`stdout`) or `OUTFILE` with no additional comments.

   Hint: Assume that plaintexts only contain lowercase, alphabetic characters and are written in English language.

5. (7 points) Implement a tool `break_mono.py` to automatically derive the key of texts encrypted with a monoalphabetic substitution cipher. Save the file in the `src/mono/` directory.

   ```
   usage: break_mono.py [-h] FILE
   ```

   The tool receives the ciphertext as positional argument and writes the derived key with no additional comments to standard output (`stdout`). Using this key it should be possible to decrypt messages using `mono.py`. Example plain- and ciphertexts are provided here:

   ```
   src/test/mono/plaintext.txt
   src/test/mono/ciphertext.txt
   ```

   *Hints: 1) Make use of the functionalities implemented above. 2) You may need a list of common text to solve this assignment. Such a list is provided under `src/test/mono/common.txt`. Please **don't** change this file. If the list isn't sufficient rethink your solution.*

6. (2 points) Implement a tool that encrypts and decrypts a message using a Vigenère cipher. The synopsis of this tool, called `vig.py`, is the same as the one for mono-alphabetic ciphers, but accepts keys of variable length rather than a fixed alphabet. Save the file in the `src/vigenere/` directory.

   ```
   usage:  vig.py [-h] (--encrypt KEY | --decrypt KEY) [--out OUTFILE ] FILE
   ```

   Inputs are pre-processed as in 4a) part i and outputs are again written to either `OUTFILE` or `stdout`. Specifying `--encrypt` encrypts the file, `--decrypt` decrypts it.

   Hint: Assume that plaintexts only contain lowercase, alphabetic characters and are written in English language.

7. (8 points) **Master:** Implement a tool `break_vig.py` to automatically derive the key of texts encrypted with a Vigenère cipher. Save the file in the `src/vigenere/` directory.

   ```
   usage:  break_vig.py [-h] --keylen INT FILE
   ```

   The tool receives the ciphertext as positional argument, the key-length as `--keylen` and writes the derived key with no additional comments to standard output (`stdout`). Using this key it should be possible to decrypt messages using `vig.py`. Example plain- and ciphertexts are provided here:

   ```
   src/vigenere/plaintext.txt
   src/vigenere/ciphertext.txt
   ```

   Hint: The plaintext has been encrypted with a ***key length of 9****.*

   Hint: You may need a list of common words to solve this assignment. Such a list is again provided under `src/vigenere/common.txt`*. Please* ***don't*** *change this file. If the list isn't sufficient rethink your solution.*