Classical Cryptography

Tobias Innleggen, András Zsolt Sütő, Azim Kazimli

Tuesday 18th March, 2025

Abstract

In this project, we investigate the effectiveness of using Haskell in implementing and breaking classical ciphers. Our report focuses on three historically significant ciphers: the Caesar cipher, the Vigenére cipher, and the Playfair cipher. We develop automated methods to recognize which cipher has been used on a given ciphertext by analyzing statistical features such as letter frequencies, n-gram patterns, entropy, and periodicity. Based on the identified cipher, we then apply appropriate cryptanalytic attacks: brute-force key search for Caesar and Playfair, and a combination of the Friedman test, Kasiski examination, and frequency analysis for Vigenére. We evaluate the success rate of these attacks and the performance of our Haskell implementations. **(Final results will be inserted here upon completion, summarizing the accuracy and efficiency achieved.)**

Contents

1 Introduction

Classical ciphers like Caesar, Vigenère, and Playfair demonstrate the rich history and fundamental principles of cryptography[BR23], while also providing an opportunity to assess modern programming paradigms. In this project, we focus on applying Haskell's functional approach to implement and break these ciphers: we design tools for automatically recognizing the encryption method (via frequency analysis, entropy metrics, and periodicity checks) and then apply tailored attacks. Notably, Vigenère is attacked using the Friedman and Kasiski tests, and Playfair is targeted through guided search and heuristic improvements.

We seek to determine how effectively functional patterns—from list comprehensions to higher-order functions—can simplify cryptographic implementation and cryptanalysis. Specifically, we measure the performance of our tools and the success rates of brute-force or frequency-based attacks, highlighting where Haskell's immutable data structures and clear abstractions streamline even the more computationally heavy tasks. The upcoming sections describe each cipher, our recognition framework, and the methods used to recover plaintext.

2 Implementing classical cryptography ciphers

Some more theory, some more citing like [**?**].

2.1 Substitution cipher

Explain code of cipher implementation in codeblocks, unhide important things from below

```
{-# LANGUAGE TypeSynonymInstances #-}
{-# LANGUAGE FlexibleInstances #-}
{-# LANGUAGE InstanceSigs #-}
module Substitution where
```

2.2 Vignere cipher

Explain code of cipher implementation in codeblocks, unhide important things from below

```
module Vignere where
```

2.3 Playfair cipher

Explain code of cipher implementation in codeblocks, unhide important things from below

```
module Playfair where
```

3 Cipher recognition

This section is about cypher recognition.

4 Attacks

In this section we perform attacks on the ciphers implemented in 2.

```
module Main where

import Vignere
import Substitution
import Playfair
import PlayfairBreaker

main :: IO ()
main = do
  putStrLn "Hello!"
```

We can show the output with

```
Hello!
Output, statistics, timings, encoded decoded results whatever:
[1,2,3,4,5,6,7,8,9,10]
[100,100,300,300,500,500,700,700,900,900]
[1,3,0,1,1,2,8,0,6,4]
[100,300,42,100,100,100,700,42,500,300]
GoodBye
```

5 Tests

We now use the library QuickCheck to randomly generate input for the ciphers, and test whether they work correctly. We can also use QuickCheck to test attacks maybe?

```
module Tests where

import Vignere
import Substitution
import Playfair

import Test.Hspec
import Test.QuickCheck

main :: IO ()
main = hspec $ do
  describe "Basics" $ do
    it "somenumbers should be the same as [1..10]" $
      somenumbers `shouldBe` [1..10]
    it "if n > - then funnyfunction n > 0" $
      property (\n -> n > 0 ==> funnyfunction n > 0)
    it "myreverse: using it twice gives back the same list" $
      property $ \str -> myreverse (myreverse str) == (str::String)
```

6 Conclusion

Here we concluded what has to be concluded.

References

[BR23] R. Banoth and R. Regar. Classical and modern cryptography for beginners. *Springer*, 2023.