Parallel Cryptanalysis of Vigenere Cipher

Alex Nicita an2582@columbia.edu

Goal

The goal of this project is to implement in Haskell a parallelized algorithm for finding the encryption keys of Vigenere ciphers [1]. Unlike the trivial task of parallelizing encryption and decryption algorithms, there is a significant challenge in decrypting a message provided only with a ciphertext [2]. In fact, these cryptanalytic algorithms often behave much more like graph search algorithms than they do traditional cryptographic algorithms, providing some intuition that parallelization may be a strong technique for performance improvements over sequential alternatives.

Background

The Vigenere cipher was invented in the 16th century and remained unbreakable for over 300 years after its original publication. Unique from the earliest known ciphers that simply mapped one letter to another, such as the Caesar cipher [3], the Vigenere cipher cannot be easily cracked with frequency analysis, which is the technique of counting letter frequencies in a message and then mapping those frequencies to letter occurrences in the alphabet being studied. Instead, the Vigenere cipher uses a repeating keyword to encode messages. This ensures that the same letter can map to different letters depending on the current index of the encoding keyword, a foundational evolution in cryptography at the time.

Vigenere Cipher Solver Algorithm

Provided with some reasonable assumptions, cracking the keys underlying Vigenere encryption is a well solved problem. For the sake of this project, we will assume that the encryption key is a word of less than 30 characters. From there, the underlying algorithm will compute the best possible fit of keys with lengths less than the maximum possible, where fit is a measure of the amount of information that is captured by checking the changes of every n-th letter. Since the ciphertext was encrypted under a repeating word which has fixed length, this procedure will consistently output a result with the encryption key.

Next Steps

The next steps for completing this project are as follows:

1. Implement a sequential algorithm for solving Vigenere ciphers
2. Generate ciphers of various length and key size for benchmarking
3. Implement parallelization techniques (parList, chunking, and more as time permits)
4. Conclude with findings, including program time completion as a function of cores

Conclusion

Over the next month, this project will seek to parallelize an algorithm for computing encryption keys provided only with a ciphertext encrypted under the Vigenere encryption algorithm. Unlike parallelizing cryptography, the haskell code written for this project will seek to parallelize cryptanalysis, which fundamentally can experience significant performance improvements through parallelization.

---

[1] Vigenere Cipher
[2] COA Security
[3] Caesar Cipher