# Vigenere Cipher Cracking and Simplified Data Encryption Standard implementation

**Abstract**

Redact this part later

## 1 Vigenere Cipher Cracking

We are given the following cipher :

```
BQZRMQ KLBOXE WCCEFL DKRYYL BVEHIZ NYJQEE BDYFJO PTLOEM EHOMIC
UYHHTS GKNJFG EHIMKN IHCTIH VRIHAR SMGQTR QCSXXC SWTNKP TMNSW
AMXVCY WEOGSR FFUEEB DKQLQZ WRKUCO FTPLOT GOJZRI XEPZSE ISXTCT
WZRMXI RIHALE SPRFAE FVYORI HNITRG PUHITM CFCDLA HIBKLH RCDIMT
WQWTOR DJCNDY YWMJCN HDUWOF DPUPNG BANULZ NGYPQU LEUXOV FFDCEE
YHQUXO YOXQUO DDCVIR RPJCAT RAQVFS AWMJCN HTSOXQ UODDAG BANURR
REZJGD VJSXOO MSDNIT RGPUHN HRSSSF VFSINH MSGPCM ZJCSLY GEWGQT
DREASV FPXEAR IMLPZW EHQGMG WSEIXE GQKPRM XIBFWL IPCHYM OTNXYV
FFDCEE YHASBA TEXCJZ VTSGBA NUDYAP IUGTLD WLKVRI HWACZG PTRYCE
VNQCUP AOSPEU KPCSNG RIHLRI KUMGFC YTDQES DAHCKP BDUJPX KPYMBD
IWDQEF WSEVKT CDDWLI NEPZSE OPYIW
```

We know that this english message has been encrypted by a polyalphabetic substitution cipher, and that the encryption key is not longer than 10 characters. To break this cipher, the first approach is to try to guess the exact key length. To do so, we are looking for repeating sequences, which have a high probability of corresponding to a repetition of the key. This method is called kasiski examination.[1]

To perform this examination, I implemented a Python method which loops through the whole cipher, and searches for patterns which appear again in the rest of the cipher. If an occurence of the current pattern is found, the distance between the two patterns is added to the possible keys list. Note that all divisors of the distance are also added to the list, as the key could have repeated multiple times between the two occurences of the pattern. The get factors submethod does not return factors which are greater than the maximum key length.

```python
def kasiski_examination(cipher):
''' Sort the key length probabilities by identifying repeated substrings '''

    possible_key_lengths = []

    for pattern_length in range(MIN_PATTERN_LENGTH, MAX_PATTERN_LENGTH):
        for index in range(len(cipher)):
            substring = cipher[index:index+pattern_length]
            distance = cipher[index+pattern_length:].find(substring)

            if distance != -1:
                possible_key_lengths.extend(get_factors(distance + pattern_length))

    probable_key_lengths = {length:possible_key_lengths.count(length) for length in
                                            possible_key_lengths}

    return sorted(probable_key_lengths.items(), key=lambda x:x[1], reverse=True)
```

When executed on the given cipher with MIN_PATTERN_LENGTH = 5 and MAX_PATTERN_LENGTH = 10, we get this output.

## 2 Simplified DES Implementation

## 3 Conclusion

## References

[1] Wikipedia contributors. Vigenre cipher — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Vigen%C3%A8re_cipher, 2019. [Online].