Chapter 1: Polyalphabetic Ciphers: Vigenere

Blaise de Vigenere (1586). Saw lots of use starting around 1800. Cracked by Chas. Babbage about 1854, not published (Crimean war secrecy?), and independently by Friedrich Wilhelm Kasiski in 1863.

Symmetric, periodic. Long thought unbreakable, known to Casanova (!!). 4-liner in Matlab.

Vigenere

Idea: repeat N-long key, plaintext underneath:

```
COMETCOMETCOMETCOMET
riverrunpasteveandad
```

Then encipher each plaintext char with shift cipher whose first letter is the key letter above.

Attacks: Chosen Plaintext: choose all a's (0), get key back. Known plaintext the same: subtract the plaintext from ciphertext (mod 26) to get effect of enciphering with 0s.

Ciphertext only: Need LCM and GCD for Kasiski (to find key length), statistics for Friedman (to find the key length **and** the key.)

Examples...

Details

Formula for Vigenere:

$E(x_i) = (x_i + k_{i\%m})\%26$, where
$k_0, k_2, ..., k_{m-1}$ is the key

$D(x_i) = (x_i - k_{i\%m})\%26$

Beaufort Variation:
$E(x_i) = D(x_i) = (k_i - x_i)\%26$.

Two encryptions with different keys yield period of LCM of key lengths. Each plaintext char can appear as more than one ciphertext char, each ciphertext char can represent more than one plaintext char. Long keys possible – one-time pad is end case.

Kasiski Attack Weakness is periodicity: exploit to find key length.

- If two identical chunks of plaintext (words, tri- or bi-grams...) are separated by some multiple of the keylength, they will generate identical chunks of ciphertext.
- Look for repeated groups of ciphertext letters, chart their separation distances.
- Factor the distances. Ideally their GCD is the key length. GCD sharpens search, not really needed.
- Not always that simple of course: look for recurring factors, any factor in most of the GCDs...
- Long messages will yield more accidents destroying GCD trick.

Why Does Kasiski Work?

- If all groups (e.g. trigrams) in English were equally likely, most matches would be accidents, no accumulation of evidence for any factor.
- Most frequent trigrams are 1-2%, but $1/(26^3)$ is .0057%, so should work for keys < 100 long.
- Work goes up as keylength squared!

Finding Key We know the key length: exploit weakness of Caesar shift

- Break ciphertext into groups of chars whose indices are the same mod m (so they were all enciphered with same shift).
- These chars are all from a rotated alphabet, so ideally, their histogram is a rotated version of the English A-Z frequency histogram.
- Find shift that maximizes correlation $\sum_{i=1}^{n} x_i * y_i$ (maximum vector dot product)
- Or which minimizes sum of absolute differences (minimum vector difference).
- If key is in meaningful, guess it after part appears.

Friedman Attack and IOC Friedman attack (1925) more systematic, based on Index of Coincidence that measures *all* forms of statistical dependence in English.

- $IOC(y, z) = (1/N) \sum \delta(y_i, z_i)$, with $\delta(a, b) = 1$ if $a = b$, else 0. Defined for any strings of same length, measures the proportion of characters that match.
- Either or both strings random: IOC = 1/26 = 3.85%.
- English is nonrandom; turns out IOC = 6.7%.
- Vigenere ciphertext is (less) nonrandom; turns out its IOC = 4.7%.
- Approximate IOC based only on single-letter frequencies: $IOC_{ave} = \sum P(y_i) * P(z_i)$. Not bad, for English it's 6.4%.

IOC Cont.

- Can compute either exact IOC (by counting) or average IOC (by computing letter frequencies and multiplying) for two specific strings.
- OR even for a single string: $IOC_{avg}(y) = \sum P(y_i)^2$
- So we've got an inner (dot) product and norm in a 26-d space of character frequencies. And dot products are good for matching.
- * If plaintexts x, y are encrypted by same substitution or transposition cipher, $IOC(E(x), E(y)) = IOC(x, y)$.
- If two strings encrypted by any (even different) transpositions, their IOC_{avg} is unchanged.

Decipherment– Key Length

Remember:

* If plaintexts x, y are encrypted by same substitution or transposition cipher, $IOC(E(x), E(y)) = IOC(x, y)$.

Thus we find key length by taking IOC of ciphertext with shifted version of itself. Look for IOC of .47 with periodic spikes of .67. Avoid shifts < 3 or 4, since local dependencies in English mask the correlation we want. May find multiple of key length due to accidents.

Decipherment– Key

Again, break ciphertext into key-length columns, each encrypted by same shift. Probably easiest at this point to use histogram-matching technique of Kasiski to find shift (key letter) in each position.

Or can take two cols i and j, reencrypt i by all the different shifts, and look for good IOC between those encryptions and j. Some small technical issues here.

If Vigenere substitution alphabets are general (not shifts), much harder, though key-length determination still works. The keys are longer of course, 26m instead of m.

Autokey

See the Assignment!