Introduction to Modern Cryptography

Michele Ciampi

(Slides courtesy of Prof. Jonathan Katz)

Lecture 2 Part 1

Vigenère Cipher

The Vigenère cipher

- **Key is a string**, not a character
- Encrypt: shift each character in the plaintext by the amount dictated by the corresponding character of the key
- Wrap around in the key as needed
- Decryption just reverses the process

```
tellhimaboutme
cafecafecafeca
veqpjiredozxoe
```

The Vigenère cipher

- Size of key space?
- Let key be **14**-character English string
- $\implies$ key space has size $\mathbf{26^{14} \approx 2^{66}}$
- Brute-force search infeasible
- Is the Vigenère cipher secure?
- (Believed secure for many years...)

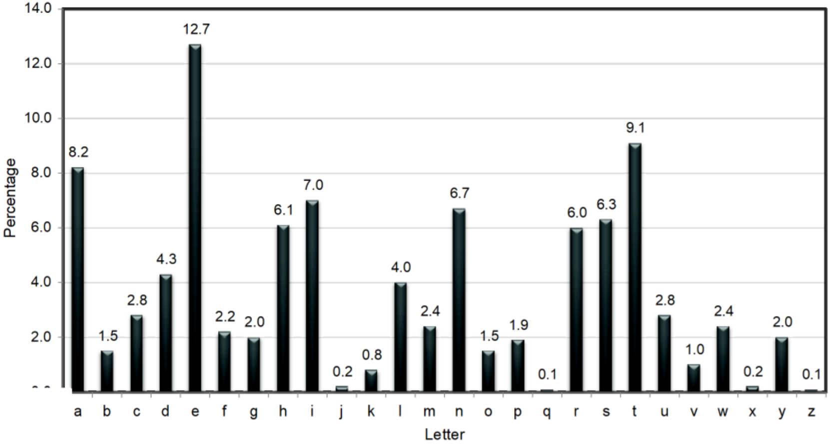
Attacking the Vigenère cipher

Observation

- Every **14**-th character is "encrypted" using the same shift
- Looking at every **14**-th character is (almost) like looking at ciphertext encrypted with the Shift Cipher
- (Direct brute-force attack still doesn't work)

```
[v]eqpjiredozxoe[u]alpcmsdjquiqn
[d]nossoscdcusoa[k]jqmxpqrhyycjq
[o]qqodhjcciowie[i]i
```

Using plaintext letter frequencies



Attacking the Vigenère cipher

- Look at every **14**-th character of the ciphertext, starting with the first – call this a ”stream”
- Let $\boldsymbol{\alpha}$ be the most common character appearing in this stream
- Most likely $\boldsymbol{\alpha}$ corresponds to the most common plaintext character i.e. $\boldsymbol{e}$
- $\implies$ guess that the first character of the key is $\boldsymbol{\alpha} - \boldsymbol{e}$
- Repeat for all other positions
- Require long ciphertext; prone to errors; can do better...

A better attack 1/2

- Let $\boldsymbol{p_i : 0 \leq i \leq 25}$ denote the frequency of the $\boldsymbol{i}$-th English letter in normal English plaintext
- Compute $\sum_i \boldsymbol{p_i}^2 = \mathbf{0.065}$: constant for English text
- Let $\boldsymbol{q_i}$ denote the **observed** frequency of the $\boldsymbol{i}$-th English letter within a given **ciphertext stream**
- ($\boldsymbol{q_i}$ is the number of times letter $\boldsymbol{i}$ appears in the ciphertext stream divided by the stream length)
- $\boldsymbol{i}$ of $\boldsymbol{q_i}$ was obtained from letter $\boldsymbol{i - j}$ for key $\boldsymbol{j}$
- Therefore $\boldsymbol{q_i \approx p_{i-j}}$ or equivalently $\boldsymbol{q_{i+j} \approx p_i}$

A better attack 2/2

- So if the key for the stream is $\boldsymbol{j}$, expect $\boldsymbol{q_{i+j} \approx p_i, \forall i}$
- So expect $\sum_i \boldsymbol{p_i q_{i+j}} \approx \mathbf{0.065}$ for the **right key** $\boldsymbol{j}$
- Test for every value of $\boldsymbol{j}$ to find the right one
- This recovers **the first key character**
- Repeat for the second stream to recover **the second key character**
- Repeat for all streams to recover **the whole key**
- **Recall:** # streams = # key characters

Finding the key length

- The previous attack assumes we know the key length
- What if we don't?
- Of course, can always try the previous attack for all possible key lengths as long as: # **key lengths** $\ll$ # **keys**
- We can do better!

Finding the key length

Observation: correct key length

- For the **correct key length**, the ciphertext frequencies $\{q_i\}$ of a stream will be shifted versions of the $\{p_i\}$
- Recall that $q_i \approx p_{i-j}$ (equivalently $q_{i+j} \approx p_i$), where j is the key (the shift)
- In other words $\{q_i\}$ is a permutation of $\{p_i\}$
- It follows that:

$$\sum_i {q_i}^2 \approx \sum_i {p_i}^2 = 0.065$$

Finding the key length

Observation: incorrect key length

- When using an **incorrect key length**, expect (heuristically) that ciphertext letters are uniform
- For uniform distribution:

$$\sum_i {q_i}^2 = \sum_i \left(\frac{1}{26}\right)^2 = 26\left(\frac{1}{26}\right)^2 = \frac{1}{26} = 0.038$$

Finding the key length

Key length recovery

- For a cadidate key length, the attacker needs to distinguish between $\sum_i q_i^2 = 0.065$ and $\sum_i q_i^2 = 0.038$
- In fact, good enough to find the key length N that maximizes $\sum_i q_i^2$
- (Can verify by looking at other streams)

Byte-wise Vigenère cipher

- The key is a **string of bytes**
- The plaintext is a **string of bytes**
- Encrypt: `XOR` each character in the plaintext with the corresponding character of the key
- Wrap around in the key as needed
- Decryption just reverses the process

Example (ASCII encoding)

- Say plaintext is **Hello!** and key is `0xA1 2F`
- **Hello!** = `0x48 65 6C 6C 6F 21` (ASCII codes)
- `XOR` with `0xA1 2F A1 2F A1 2F`
- `0x48` $\oplus$ `0xA1`
  - `0100 1000` $\oplus$ `1010 0001` = `1110 1001` = `0xE9`
- Ciphertext: `0xE9 4A CD 43 CE 0E`

Attacking the Byte-wise Vigenère cipher

Two steps of the attack

1. Determine the key length
2. Determine each byte of the key

- Let $\boldsymbol{p_i : \ 0 \leq i \leq 255}$ – frequency of byte $\boldsymbol{i}$ in normal English (ASCII) plaintext
- e.g. $\boldsymbol{p_{97}}$ = frequency of $\boldsymbol{a}$ (**97** is ASCII for $\boldsymbol{a}$)
- Note that $\boldsymbol{p_i = 0 : \ \forall i < 32, \ \forall i > 127}$
- If $\boldsymbol{\{p_i\}}$ are known, use same principles as before
- What if $\boldsymbol{\{p_i\}}$ are not known?

Step **1**: Determining the key length

- Let $\boldsymbol{N}$ – correct key length; $\boldsymbol{M}$ – any incorrect key length
- Every $\boldsymbol{N}$-th character of plaintext is encrypted using the same key byte (”shift”)
- If we take every $\boldsymbol{N}$-th character and calculate $\boldsymbol{\{q_i\}}$, we get the set $\boldsymbol{\{p_i\}}$ in permuted order
- If we take every $\boldsymbol{M}$-th character and calculate $\boldsymbol{\{q_i\}}$, we get something close to uniform
- **Observe:** we **don’t** need to know $\boldsymbol{\{p_i\}}$ to distinguish these two!

Step 1: Determining the key length

- For some candidate key length, tabulate $\{q_0, \ldots, q_{255}\}$ for the first stream (say) and compute $\sum_i {q_i}^2$
- If close to uniform:

$$\sum_i {q_i}^2 \approx 256(\frac{1}{256})^2 = \frac{1}{256}$$

- If a permutation of $\{p_i\}$:

$$\sum_i {q_i}^2 \approx \sum_i {p_i}^2 \gg \frac{1}{256}$$

Step 1: Determining the key length

- Key point: for correct length, $\sum_i q_i^2$ much larger than $\frac{1}{256}$
- So compute $\sum_i q_i^2$ for each possible key length, and look for maximum value
- Correct key length N should yield a large value **for all N streams**

Step **2**: Determining the $\boldsymbol{i}$-th byte of the key

- Assume the key length $\boldsymbol{N}$ is known
- Look at $\boldsymbol{i}$-th ciphertext stream
- As before, all bytes in this stream were generated by `XOR` -ing plaintext with **the same byte of the key** (the $\boldsymbol{i}$-th byte!)
- Decrypt the stream with every possible byte value $\boldsymbol{B}$
- Get a candidate plaintext stream for each value
- i.e. **256** decrypted candidate plaintext streams

Step $\mathbf{2}$: Determining the $\boldsymbol{i}$-th byte of the key

If guess for $\boldsymbol{B}$ is correct

- All bytes in plaintext stream will be between $\mathbf{32}$ and $\mathbf{126}$
- Frequency of space character should be high
- Frequencies of lowercase letters (as a fraction of all lowercase letters) should be close to known English-letter frequencies:
  - Tabulate observed letter frequencies $\boldsymbol{q'_0 \ldots q'_{25}}$ (as fraction of all lowercase letters) in the candidate plaintext
  - Should find $\sum \boldsymbol{q'_i p'_i} \approx \sum \boldsymbol{p'^2_i} \approx \mathbf{0.065}$, where $\boldsymbol{p'_i}$ corresponds to English-letter frequencies
- In practice, take $\boldsymbol{B}$ that maximizes $\sum \boldsymbol{q'_i p'_i}$

Attack time?

Time for determining the key length

- Let the key length be at most $\boldsymbol{L}$ i.e. $\mathbf{1} \leq \boldsymbol{N} \leq \boldsymbol{L}$
- Execute at most $\boldsymbol{L}$ trials for the correct key length
  - In each trial compute $\mathbf{256}$ frequencies $\boldsymbol{q_i} : \ \mathbf{0} \leq \boldsymbol{i} \leq \mathbf{255}$
- Total time: $\approx \mathbf{256}\ \boldsymbol{L}$

Attack time?

Time for determining the key

- To deterime the i-th byte of the key:
  - Execute $\mathbf{256}$ decryptions of the i-th stream for each candidate value B
    - In each decryption compute $\mathbf{256}$ frequencies $q_i' : \mathbf{0} \leq i \leq \mathbf{255}$
- Total time to recover the i-th byte: $\approx \mathbf{256^2}$
- Total time to recover all key bytes: $\leq \mathbf{256^2} L$

Time for Brute-force

$$\mathbf{256}^L$$

Total attack time vs. brute-force

$$\mathbf{256}\boldsymbol{L} + \mathbf{256^2}\boldsymbol{L} \approx \mathbf{256^2}\boldsymbol{L} \ll \mathbf{256}^{\boldsymbol{L}}$$

Note

The attack is more reliable as the ciphertext length grows larger

Lessons learned

Crypto Design Lesson One (recall)

- The key space must be large enough to make brute-force attacks impractical (cf. Shift Cipher)

Crypto Design Lesson Two

- Large key space is a necessary, but not sufficient condition for a secure encryption scheme (cf. Vigenère Cipher)

But what does *secure* actually mean? (next lecture!)

End

Reference: Section 1.3 of the book