Classic Ciphers I

Lectore 02
Byte-wise Shift Cipher
Byte-wise Shift Cipher

- Instead of $a, b, c, d, \ldots, z$ have (for example) 0000, 0001, $\ldots$, 1111.

- Works for an alphabet of *bytes* rather than (English, lowercase) *letters*
  
  - Data in a computer is stored this way anyway. So works natively for arbitrary data!

- Use XOR instead of modular addition. Fast!
- Decode and Encode are both XOR.
  
  - Essential properties still hold
# Hexadecimal (base 16)

<table>
<thead>
<tr>
<th>Hex</th>
<th>Bits (&quot;nibble&quot;)</th>
<th>Decimal</th>
<th>Hex</th>
<th>Bits (&quot;nibble&quot;)</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>2</td>
<td>A</td>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>3</td>
<td>B</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>4</td>
<td>C</td>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>5</td>
<td>D</td>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>6</td>
<td>E</td>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>7</td>
<td>F</td>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>
Hexadecimal (base 16)

**Notation:** 0x before a string of \{0, 1, \ldots, 9, A, B, C, D, E, F\} means that the string will be base 16.

- **0x10**
  - 0x10 = 16*1 + 0 = 16
  - 0x10 = 0001 0000

- **0xAF**
  - 0xAF = 16*A + F = 16*10 + 15 = 175
  - 0xAF = 1010 1111
Characters (often) represented in ASCII with TWO hex-digits.

Potentially 256 characters via
\[ \{0, \ldots, 9, A, \ldots, F\} \times \{0, \ldots, 9, A, \ldots, F\} \]

Only use 128 characters via \[ \{0, \ldots, 8\} \times \{0, \ldots, 9, A, \ldots, F\} \]
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Char</th>
<th>Hex</th>
<th>Dec</th>
<th>Char</th>
<th>Hex</th>
<th>Dec</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0</td>
<td>NULL</td>
<td>0x20</td>
<td>32</td>
<td>Space</td>
<td>0x40</td>
<td>64</td>
<td>@</td>
</tr>
<tr>
<td>0x01</td>
<td>1</td>
<td>SOH</td>
<td>0x21</td>
<td>33</td>
<td>!</td>
<td>0x41</td>
<td>65</td>
<td>A</td>
</tr>
<tr>
<td>0x02</td>
<td>2</td>
<td>STX</td>
<td>0x22</td>
<td>34</td>
<td>&quot;</td>
<td>0x42</td>
<td>66</td>
<td>B</td>
</tr>
<tr>
<td>0x03</td>
<td>3</td>
<td>ETX</td>
<td>0x23</td>
<td>35</td>
<td>#</td>
<td>0x43</td>
<td>67</td>
<td>C</td>
</tr>
<tr>
<td>0x04</td>
<td>4</td>
<td>EOT</td>
<td>0x24</td>
<td>36</td>
<td>$</td>
<td>0x44</td>
<td>68</td>
<td>D</td>
</tr>
<tr>
<td>0x05</td>
<td>5</td>
<td>ENQ</td>
<td>0x25</td>
<td>37</td>
<td>%</td>
<td>0x45</td>
<td>69</td>
<td>E</td>
</tr>
<tr>
<td>0x06</td>
<td>6</td>
<td>ACK</td>
<td>0x26</td>
<td>38</td>
<td>&amp;</td>
<td>0x46</td>
<td>70</td>
<td>F</td>
</tr>
<tr>
<td>0x07</td>
<td>7</td>
<td>BELL</td>
<td>0x27</td>
<td>39</td>
<td>'</td>
<td>0x47</td>
<td>71</td>
<td>G</td>
</tr>
<tr>
<td>0x08</td>
<td>8</td>
<td>BS</td>
<td>0x28</td>
<td>40</td>
<td>(</td>
<td>0x48</td>
<td>72</td>
<td>H</td>
</tr>
<tr>
<td>0x09</td>
<td>9</td>
<td>TAB</td>
<td>0x29</td>
<td>41</td>
<td>)</td>
<td>0x49</td>
<td>73</td>
<td>I</td>
</tr>
<tr>
<td>0x0A</td>
<td>10</td>
<td>LF</td>
<td>0x2A</td>
<td>42</td>
<td>*</td>
<td>0x4A</td>
<td>74</td>
<td>J</td>
</tr>
<tr>
<td>0x0B</td>
<td>11</td>
<td>VT</td>
<td>0x2B</td>
<td>43</td>
<td>+</td>
<td>0x4B</td>
<td>75</td>
<td>K</td>
</tr>
<tr>
<td>0x0C</td>
<td>12</td>
<td>FF</td>
<td>0x2C</td>
<td>44</td>
<td>,</td>
<td>0x4C</td>
<td>76</td>
<td>L</td>
</tr>
<tr>
<td>0x0D</td>
<td>13</td>
<td>CR</td>
<td>0x2D</td>
<td>45</td>
<td>-</td>
<td>0x4D</td>
<td>77</td>
<td>M</td>
</tr>
<tr>
<td>0x0E</td>
<td>14</td>
<td>SO</td>
<td>0x2E</td>
<td>46</td>
<td>.</td>
<td>0x4E</td>
<td>78</td>
<td>N</td>
</tr>
<tr>
<td>0x0F</td>
<td>15</td>
<td>SI</td>
<td>0x2F</td>
<td>47</td>
<td>/</td>
<td>0x4F</td>
<td>79</td>
<td>O</td>
</tr>
<tr>
<td>0x10</td>
<td>16</td>
<td>DLE</td>
<td>0x30</td>
<td>48</td>
<td>0</td>
<td>0x50</td>
<td>80</td>
<td>P</td>
</tr>
<tr>
<td>0x11</td>
<td>17</td>
<td>DC1</td>
<td>0x31</td>
<td>49</td>
<td>1</td>
<td>0x51</td>
<td>81</td>
<td>Q</td>
</tr>
<tr>
<td>0x12</td>
<td>18</td>
<td>DC2</td>
<td>0x32</td>
<td>50</td>
<td>2</td>
<td>0x52</td>
<td>82</td>
<td>R</td>
</tr>
<tr>
<td>0x13</td>
<td>19</td>
<td>DC3</td>
<td>0x33</td>
<td>51</td>
<td>3</td>
<td>0x53</td>
<td>83</td>
<td>S</td>
</tr>
<tr>
<td>0x14</td>
<td>20</td>
<td>DC4</td>
<td>0x34</td>
<td>52</td>
<td>4</td>
<td>0x54</td>
<td>84</td>
<td>T</td>
</tr>
<tr>
<td>0x15</td>
<td>21</td>
<td>NAK</td>
<td>0x35</td>
<td>53</td>
<td>5</td>
<td>0x55</td>
<td>85</td>
<td>U</td>
</tr>
<tr>
<td>0x16</td>
<td>22</td>
<td>SYN</td>
<td>0x36</td>
<td>54</td>
<td>6</td>
<td>0x56</td>
<td>86</td>
<td>V</td>
</tr>
<tr>
<td>0x17</td>
<td>23</td>
<td>ETB</td>
<td>0x37</td>
<td>55</td>
<td>7</td>
<td>0x57</td>
<td>87</td>
<td>W</td>
</tr>
<tr>
<td>0x18</td>
<td>24</td>
<td>CAN</td>
<td>0x38</td>
<td>56</td>
<td>8</td>
<td>0x58</td>
<td>88</td>
<td>X</td>
</tr>
<tr>
<td>0x19</td>
<td>25</td>
<td>EM</td>
<td>0x39</td>
<td>57</td>
<td>9</td>
<td>0x59</td>
<td>89</td>
<td>Y</td>
</tr>
<tr>
<td>0x1A</td>
<td>26</td>
<td>SUB</td>
<td>0x3A</td>
<td>58</td>
<td>:)</td>
<td>0x5A</td>
<td>90</td>
<td>Z</td>
</tr>
<tr>
<td>0x1B</td>
<td>27</td>
<td>FSC</td>
<td>0x3B</td>
<td>59</td>
<td>;</td>
<td>0x5B</td>
<td>91</td>
<td>[</td>
</tr>
<tr>
<td>0x1C</td>
<td>28</td>
<td>FS</td>
<td>0x3C</td>
<td>60</td>
<td>&lt;</td>
<td>0x5C</td>
<td>92</td>
<td>\</td>
</tr>
<tr>
<td>0x1D</td>
<td>29</td>
<td>GS</td>
<td>0x3D</td>
<td>61</td>
<td>=</td>
<td>0x5D</td>
<td>93</td>
<td>]</td>
</tr>
<tr>
<td>0x1E</td>
<td>30</td>
<td>RS</td>
<td>0x3E</td>
<td>62</td>
<td>&gt;</td>
<td>0x5E</td>
<td>94</td>
<td>^</td>
</tr>
<tr>
<td>0x1F</td>
<td>31</td>
<td>US</td>
<td>0x3F</td>
<td>63</td>
<td>?</td>
<td>0x5F</td>
<td>95</td>
<td>_</td>
</tr>
</tbody>
</table>
ASCII

- ‘1’ = 0x31 = 0011 0001
- ‘F’ = 0x46 = 0100 0110
Useful observations

- Only 128 valid ASCII chars (128 bytes invalid)
- \(0x20-0x7E\) printable
- \(0x41-0x7A\) includes upper/lowercase letters
  - Uppercase letters begin with \(0x4\) or \(0x5\)
  - Lowercase letters begin with \(0x6\) or \(0x7\)
Byte-wise shift cipher

- $\mathcal{M} = \{\text{strings of bytes}\}$

- $Gen$: choose uniform byte $k \in \mathcal{K} = \{0, \ldots, 255\}$

- $Enc_k(m_1 \ldots m_t)$: output $c_1 \ldots c_t$, where $c_i := m_i \oplus k$

- $Dec_k(c_1 \ldots c_t)$: output $m_1 \ldots m_t$, where $m_i := c_i \oplus k$

- Verify that correctness holds...
Example

Key is 11001110.
Alice wants to send 00011010, 11100011, 00000000
She sends

00011010 ⊕ 11001110, 11100011 ⊕ 11001110, 00000000 ⊕ 11001110

= 11010100, 00101101, 11001110

Question: Should it worry Alice and Bob that the key itself was transmitted? Discuss
No. Eve has no way of knowing that.
Example

Key is \texttt{11001110}.
Alice wants to send \texttt{00011010, 11100011, 00000000}
She sends

\texttt{00011010 \oplus 11001110, 11100011 \oplus 11001110, 00000000 \oplus 11001110}

\[= \texttt{11010100, 00101101, 11001110}\]

Question: Should it worry Alice and Bob that the key itself was transmitted? Discuss
Example

Key is 11001110.
Alice wants to send 00011010, 11100011, 00000000
She sends

00011010 \oplus 11001110, 11100011 \oplus 11001110, 00000000 \oplus 11001110

= 11010100, 00101101, 11001110

Question: Should it worry Alice and Bob that the key itself was transmitted? Discuss
No. Eve has no way of knowing that.
Is this cipher secure?

- No – only 256 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - If ciphertext is long enough, only one plaintext will “look like English” (use the vector method of the last set of slides).

- Can further optimize
  - First nibble of plaintext likely 0x4, 0x5, 0x6, 0x7 (assuming letters only)
  - Can reduce exhaustive search to 26 keys (how?)
  - Talk to your friends or blood enemies about this.
Sufficient key space principle

- The key space must be large enough to make exhaustive-search attacks impractical
  - How large do you think that is?

- Note: this makes some assumptions...
  - English-language plaintext
  - Ciphertext sufficiently long so only one valid plaintext
Is this cipher secure if we are transmitting numbers?

If Alice sends Bob a Document in English via Byte-Shift then insecure!

What if Alice sends Bob a credit card number? Discuss
Is this cipher secure if we are transmitting numbers?

If Alice sends Bob a Document in English via Byte-Shift then insecure!

What if Alice sends Bob a credit card number? Discuss Credit Card Numbers also have patterns:

1. Visa cards always begin with 4
2. American Express always begins 34 or 37
3. Mastercard starts with 51 or 52 or 53 or 54.

Upshot: If Eve knows what kind of information is being transmitted (English, Credit Card Numbers, numbers on checks) she can use this to make any cipher with a small key space insecure.
Affine, Quadratic, Cubic, and Polynomial Ciphers
Affine Cipher

**Recall:** Shift cipher with shift $s$:

1. Encrypt via $x \rightarrow x + s \pmod{26}$.
2. Decrypt via $x \rightarrow x - s \pmod{26}$.

We replace $x + s$ with more elaborate functions.

**Definition:** The Affine cipher with $a, b$:

1. Encrypt via $x \rightarrow ax + b \pmod{26}$.
2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$

---

**Does this work? Vote YES or NO or OTHER**

Answer: OTHER

$2x + 1$ does not work: 0 and 13 both map to 1.

Need the map to be a bijection so it will have a unique inverse.

Condition on $a, b$ so that $x \rightarrow ax + b$ is a bijection:

- $a$ is rel prime to 26.

Condition on $a, b$ so that $a$ has an inv mod 26:

- $a$ is rel prime to 26.
Affine Cipher

Recall: Shift cipher with shift $s$:

1. Encrypt via $x \rightarrow x + s \pmod{26}$.
2. Decrypt via $x \rightarrow x - s \pmod{26}$.

We replace $x + s$ with more elaborate functions.

Definition: The Affine cipher with $a, b$:

1. Encrypt via $x \rightarrow ax + b \pmod{26}$.
2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$

Does this work? Vote YES or NO or OTHER

2 $x + 1$ does not work: 0 and 13 both map to 1.

Need the map to be a bijection so it will have a unique inverse.

Condition on $a, b$ so that $x \rightarrow ax + b$ is a bij:

$a$ rel prime to 26.

Condition on $a, b$ so that $a$ has an inv mod 26:

$a$ rel prime to 26.
Affine Cipher

Recall: Shift cipher with shift $s$:

1. Encrypt via $x \rightarrow x + s \pmod{26}$.
2. Decrypt via $x \rightarrow x - s \pmod{26}$.

We replace $x + s$ with more elaborate functions

Definition: The Affine cipher with $a, b$:

1. Encrypt via $x \rightarrow ax + b \pmod{26}$.
2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$

Does this work? Vote YES or NO or OTHER Answer: OTHER

$2x + 1$ does not work: 0 and 13 both map to 1.

Condition on $a, b$ so that $x \rightarrow ax + b$ is a bijection:

$a$ is rel prime to 26.

Condition on $a, b$ so that $a$ has an inv mod 26:

$a$ is rel prime to 26.
Affine Cipher

Recall: Shift cipher with shift $s$:
1. Encrypt via $x \rightarrow x + s \pmod{26}$.
2. Decrypt via $x \rightarrow x - s \pmod{26}$.

We replace $x + s$ with more elaborate functions

**Definition:** The Affine cipher with $a, b$:
1. Encrypt via $x \rightarrow ax + b \pmod{26}$.
2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$

Does this work? Vote YES or NO or OTHER

Answer: OTHER

$2x + 1$ does not work: 0 and 13 both map to 1.
**Affine Cipher**

**Recall:** Shift cipher with shift $s$:
1. Encrypt via $x \rightarrow x + s \pmod{26}$.
2. Decrypt via $x \rightarrow x - s \pmod{26}$.

We replace $x + s$ with more elaborate functions

**Definition:** The Affine cipher with $a, b$:
1. Encrypt via $x \rightarrow ax + b \pmod{26}$.
2. Decrypt via $x \rightarrow a^{-1}(x - b) \pmod{26}$

Does this work? Vote YES or NO or OTHER Answer: OTHER

$2x + 1$ does not work: 0 and 13 both map to 1.

Need the map to be a bijection so it will have a unique inverse.
Affine Cipher

Recall: Shift cipher with shift \( s \):
1. Encrypt via \( x \rightarrow x + s \pmod{26} \).
2. Decrypt via \( x \rightarrow x - s \pmod{26} \).

We replace \( x + s \) with more elaborate functions

Definition: The Affine cipher with \( a, b \):
1. Encrypt via \( x \rightarrow ax + b \pmod{26} \).
2. Decrypt via \( x \rightarrow a^{-1}(x - b) \pmod{26} \)

Does this work? Vote YES or NO or OTHER

Answer: OTHER

\( 2x + 1 \) does not work: 0 and 13 both map to 1.

Need the map to be a bijection so it will have a unique inverse.

Condition on \( a, b \) so that \( x \rightarrow ax + b \) is a bij: \( a \) rel prime to 26.
Condition on \( a, b \) so that \( a \) has an inv mod 26: \( a \) rel prime to 26.
Shift vs Affine

**Shift:** Key space is size 26

**Affine:** Key space is
\[|\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}| \times 26 = 12 \times 26 = 312\]

*In an Earlier Era* Affine would be harder to crack than Shift.
**Shift vs Affine**

**Shift:** Key space is size 26

**Affine:** Key space is
\[|\{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}| \times 26 = 12 \times 26 = 312\]

In an Earlier Era Affine would be harder to crack than Shift.

Today They are both easy to crack.

Both Need: The Is English algorithm. Reading through 312 transcripts to see which one looks like English would take A LOT of time!
The Quadratic Cipher

**Definition**: The Quadratic cipher with $a$, $b$, $c$:

1. Encrypt via $x \rightarrow ax^2 + bx + c \pmod{26}$.
The Quadratic Cipher

Definition: The Quadratic cipher with $a, b, c$:
1. Encrypt via $x \rightarrow ax^2 + bx + c \pmod{26}$.

Does this work? Vote YES or NO
The Quadratic Cipher

**Definition:** The Quadratic cipher with $a, b, c$:

1. Encrypt via $x \rightarrow ax^2 + bx + c \pmod{26}$.

Does this work? Vote YES or NO Answer: NO
The Quadratic Cipher

Definition: The Quadratic cipher with $a$, $b$, $c$:

1. Encrypt via $x \rightarrow ax^2 + bx + c \pmod{26}$.

Does this work? Vote YES or NO Answer: NO

1. No easy test for Invertibility (depends on def of easy).
2. It turns out that every quadratic function mod 26 is an affine function.
The Polynomial Cipher

**Definition:** Poly Cipher with poly $p$ (coefficients in $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow p(x) \pmod{26}$.
2. Decrypt via $x \rightarrow p^{-1}(x) \pmod{26}$.

Given a polynomial over mod 26 (or any mod) does it have an inverse? What is the complexity of this problem?  
**Vote:** P, NP-complete, unknown to science.
The Polynomial Cipher

**Definition:** Poly Cipher with poly $p$ (coefficients in $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow p(x) \pmod{26}$.
2. Decrypt via $x \rightarrow p^{-1}(x) \pmod{26}$.

Given a polynomial over mod 26 (or any mod) does it have an inverse? What is the complexity of this problem?

**Vote:** P, NP-complete, unknown to science.

Unknown to Science but if over mod a prime then in P.
The Polynomial Cipher

**Definition:** Poly Cipher with poly $p$ (coefficients in $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow p(x) \pmod{26}$.
2. Decrypt via $x \rightarrow p^{-1}(x) \pmod{26}$.

Given a polynomial over mod 26 (or any mod) does it have an inverse? What is the complexity of this problem?

**Vote:** P, NP-complete, unknown to science.

**Unknown to Science** but if over mod a prime then in P.

Course website, **Notes**, has pointer to blog of mine on this. Some of the comments have theorems and pointers to the literature.
The Polynomial Cipher

**Definition:** Poly Cipher with poly \( p \) (coefficients in \( \{0, \ldots, 25\} \).

1. Encrypt via \( x \rightarrow p(x) \pmod{26} \).
2. Decrypt via \( x \rightarrow p^{-1}(x) \pmod{26} \).

Given a polynomial over mod 26 (or any mod) does it have an inverse? What is the complexity of this problem?

**Vote:** P, NP-complete, unknown to science.

**Unknown to Science** but if over mod a prime then in P.

Course website, **Notes**, has pointer to blog of mine on this. Some of the comments have theorems and pointers to the literature.

The first place The Polynomial Cipher appeared was
The Polynomial Cipher

**Definition:** Poly Cipher with poly $p$ (coefficients in $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow p(x) \pmod{26}$.
2. Decrypt via $x \rightarrow p^{-1}(x) \pmod{26}$.

Given a polynomial over mod 26 (or any mod) does it have an inverse? What is the complexity of this problem?

**Vote:** P, NP-complete, unknown to science.

Unknown to Science but if over mod a prime then in P.

Course website, Notes, has pointer to blog of mine on this. Some of the comments have theorems and pointers to the literature.

The first place The Polynomial Cipher appeared was my 3-week summer course on crypto for High School Students.

So, as the kids say, its not a thing.
General Substitution Cipher

Shift and Affine were good for Alice and Bob since

1. Easy to encrypt, Easy to decrypt
2. Short Key: Roughly 5 bits for Shift, 10 bits for Affine.

**Definition:** Gen Sub Cipher with perm $f$ on $\{0, \ldots, 25\}$.

1. Encrypt via $x \to f(x)$.
2. Decrypt via $x \to f^{-1}(x)$

1. Key is now permutation, roughly 125 bits.
2. Encrypt and Decrypt slightly harder

Uncrackable! Eve has to go through all $26!$ possibilities!!

NOT EVEN CLOSE! Eve can use Freq Analysis
General Substitution Cipher

Shift and Affine were good for Alice and Bob since

1. Easy to encrypt, Easy to decrypt
2. Short Key: Roughly 5 bits for Shift, 10 bits for Affine.

**Definition:** Gen Sub Cipher with perm \( f \) on \( \{0, \ldots, 25\} \).

1. Encrypt via \( x \rightarrow f(x) \).
2. Decrypt via \( x \rightarrow f^{-1}(x) \)

1. Key is now permutation, roughly 125 bits.
2. Encrypt and Decrypt slightly harder

**Uncrackable!** Eve has to go through all 26! possibilities!!
Shift and Affine were good for Alice and Bob since

1. Easy to encrypt, Easy to decrypt
2. Short Key: Roughly 5 bits for Shift, 10 bits for Affine.

**Definition:** Gen Sub Cipher with perm $f$ on $\{0, \ldots, 25\}$.

1. Encrypt via $x \rightarrow f(x)$.
2. Decrypt via $x \rightarrow f^{-1}(x)$

1. Key is now permutation, roughly 125 bits.
2. Encrypt and Decrypt slightly harder

**Uncrackable!** Eve has to go through all $26!$ possibilities!!

**NOT EVEN CLOSE!** Eve can use Freq Analysis
Alice sends Bob a LONG text encrypted by Gen Sub Cipher. Eve finds freq of letters, pairs, triples, ....

Text in English.

1. Can use known freq: e is most common letter, th is most common pair.

2. If Alice is telling Bob about Mid East Politics than may need to adjust: q is more common (Iraq, Qatar) and some words more common.
Pangrams: Sentence where each letter occurs at least once.

Short Panagrams ruin Freq analysis. Here are some:

1. The quick brown fox jumps over the lazy dog.
2. Pack my box with five dozen liquor jugs.
3. Amazingly few discotheques provide jukeboxes.
4. Watch Jeopardy! Alex Trebek’s fun TV quiz game.
Silly Counter Example – Lipograms

Lipograms: A work that omits one letter

1. Gadsby is a 50,000-word novel with no e.
2. Eunoia is a 5-chapter novel, indexed by vowels. Chapter A only use the vowel A, etc.
3. How I met your mother, Season 9, Episode 9: Lily and Robin challenge Barney to get a girl’s phone number without using the letter e.

We are not going to deal with this silyness!
We assume long normal texts!
Alternatives to Gen Sub (History)

In the Year 2018 Alice can easily generate a random permutation of \{a, \ldots, z\} and send it to Bob.

In the Year 1018 Alice needs a way to encode a random-looking permutation of \{a, \ldots, z\} and transmit it to Bob. So need SHORT description of random-looking perm.

1. Two ways to do this will be on the HW.
2. Foreshadowing the need for a short description of a random-looking string of bits which we will be central later in this course.
The Vigenére Cipher
EDUCATION NOTE: In class we started but did not finish Vig Cipher. I include everything on Vig Cipher in both this set of slides and the next.
The Vigenère cipher

**Key**: A word or phrase. Example: \( \text{dog} = (3,14,6) \).
Easy to remember and transmit.

**Example** using \( \text{dog} \).
Shift 1st letter by 3
Shift 2nd letter by 14
Shift 3rd letter by 6
Shift 4th letter by 3
Shift 5th letter by 14
Shift 6th letter by 6, etc.

*Jacob Prinz is a Physics Major*

*Jacob prinz isaph ysics major*

encrypts to

*MOIRP VUWTC WYDDN BGOFG SDXUU*
The Vigenère cipher

Key: \( k = (k_1, k_2, \ldots, k_n). \)
Encrypt (all arithmetic is mod 26)

\[
Enc(m_1, m_2, \ldots, m_N) = m_1 + k_1, m_2 + k_2, \ldots, m_n + k_n,
\]

\[
m_{n+1} + k_1, m_{n+2} + k_2, \ldots, m_{n+n} + k_n,
\]

\[
\ldots
\]

Decrypt Decryption just reverse the process
The Vigenère cipher

- Size of key space?
  - If keys are 14-char then key space size $26^{14} \approx 2^{66}$
  - If variable length keys, even more.
  - Brute-force search infeasible

- Is the Vigenère cipher secure?
  - Believed secure for many years...
  - Might not have even been secure then...
Cracking Vig cipher: Step One-find Keylength

Assume $T$ is a text encoded by Vig, key length $L$ unknown. For $0 \leq i \leq L - 1$, letters in pos $\equiv i \pmod{26}$ – same shift. Look for a sequence of (say) 3-letters to appear (say) 4 times.

Example: aiq appears in the
57-58-59th slot, 87-88-89th slot 102-103-104th slot
162-163-164th slot

Important: Very likely that aiq encrypted the same 3-letter sequence and hence the length of the key is a divisor of 87-57=30 102-87=15 162-102=60
The only possible $L$’s are 1,3,5,15.

Good Enough: We got the key length down to a small finite set.
Important Point about letter Freq

Assume (and its roughly true): In an English text of length $N$:

- $e$ occurs $\sim 13\%$
- $t$ occurs $\sim 9\%$
- $a$ occurs $\sim 8\%$

Etc- other letters have frequencies that are true for all texts.
Important Point about letter Freq

Assume (and its roughly true): In an English text of length $N$:

- $e$ occurs $\sim 13\%$
- $t$ occurs $\sim 9\%$
- $a$ occurs $\sim 8\%$

Etc- other letters have frequencies that are true for all texts.

Assume (and its roughly true): In an English text of length $N$, if $i \ll N$, then if you take every $i$th letter of $T$:

- $e$ occurs $\sim 13\%$
- $t$ occurs $\sim 9\%$
- $a$ occurs $\sim 8\%$

Etc- other letters same frequencies as normal texts.
Important Point about letter Freq

Assume (and its roughly true): In an English text of length $N$:

- $e$ occurs $\sim 13\%$
- $t$ occurs $\sim 9\%$
- $a$ occurs $\sim 8\%$

Etc- other letters have frequencies that are true for all texts.

Assume (and its roughly true): In an English text of length $N$, if $i \ll N$, then if you take every $i$th letter of $T$:

- $e$ occurs $\sim 13\%$
- $t$ occurs $\sim 9\%$
- $a$ occurs $\sim 8\%$

Etc- other letters same frequencies as normal texts.

Relevant to us:

- $\bar{q}$ freq of every $L$th letter: then $\sum_{i=1}^{26} q_i^2 \approx 0.065$.
- $\bar{q}$ is NOT (we won’t define that rigorously): $\sum_{i=1}^{26} q_i^2$ MUCH lower.
**Cracking Vig cipher: Step One-find Keylength**

Let $K$ be the set of possible key lengths. $K$ is small. For every $L \in K$:

- Form a stream of every $L$th character.
- Find the frequencies of that stream: $\vec{q}$.
- Compute $Q = \sum q_i^2$
- If $Q \approx 0.065$ then YES $L$ is key length.
- If $Q$ much less than 0.065 then NO $L$ is not key length.
- One of these two will happen
- Just to make sure, check another stream.

**Note:** Differs from Is English:

*Is English* wanted to know if the text was actually English. What we do above is see if the text has same dist of English, but okay if diff letters. E.g., if $z$ is 13%, $a$ is 9%, and other letters have roughly same numbers as English then we know the stream is SOME Shift. We later use *Is English* to see which shift.
A Note on Finding Keylength

We presented one method:

1. Find phrase of length $x$ appearing $y$ times. Differences $D$.
2. $K$ is set of divisors of all $L \in D$. Correct keylength in $K$.
3. Test $L \in K$ for key length until find one that works.

Alternative just try all key lengths up to a certain length:

1. Let $K = \{1, \ldots, 100\}$ (I am assuming key length $\leq 100$).
2. Test $L \in K$ for key length until find one that works.

Note: With modern computers use Method 2. In days of old eyeballing it made method 1 reasonable.
Cracking the Vig cipher: Step Two-Freq Anal

After Step One we have the key length $L$. Note:

- Every $L^{th}$ character is “encrypted” using the same shift.
- **Important**: Letter Freq still hold if you look at every $L$ 14th letter!

Step Two:

1. Separate text $T$ into $L$ streams depending on position mod $L$
2. For each steam try every shift and use Is English to determine which shift is correct.
3. You now know all shifts for all positions. Decrypt!
Using plaintext letter frequencies

![Bar chart showing letter frequencies with 'e' at 12.7%, 't' at 9.1%, 'i' at 7.0%, 'a' at 8.2%, 'h' at 6.1%, 'o' at 6.0%, 'n' at 6.7%, 'r' at 6.3%, 's' at 4.0% and 'w' at 2.4%.]
Byte-wise Vigenère cipher

- The key is a string of bytes
- The plaintext is a string of bytes
- To encrypt, XOR each character in the plaintext with the next character of the key
  - Wrap around in the key as needed
- Decryption just reverses the process.

Note: Decryption and Encryption both use XOR with same key.
Note: Can be cracked as original Vig can be cracked.