Introduction to Modern Cryptography

Michele Ciampi

(Slides courtesy of Prof. Jonathan Katz)

Lecture 01 Part 3
Shift Cipher
The Shift Cipher

- Consider encrypting English text
- Associate $a \leftarrow 0$, $b \leftarrow 1$, ..., $z \leftarrow 25$
  - $k \in \mathcal{K} = \{0, \ldots, 25\}$
- To encrypt using key $k$, shift every letter of the plaintext by $k$ positions (with wraparound)
- Decryption just does the reverse

helloworldz
ccccccccccccc
jgnnqyqtnfb
Modular arithmetic

- $x = y \pmod{N}$ if and only if $N$ divides $x - y$
  - $[x \pmod{N}] = \text{the remainder when } x \text{ is divided by } N$
  - i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x = y \pmod{N}$

- $25 = 35 \mod 10$
- $25 \neq [35 \mod 10]$
- $5 = [35 \mod 10]$
The Shift Cipher, formally

- $\mathcal{M} = \text{strings over lowercase English alphabet}$
- $\text{Gen}: \text{choose uniform } k \in \{0, \ldots, 25\}$
- $\text{Enc}_k(m_1 \ldots m_t): \text{output } c_1 \ldots c_t, \text{ where }$
  \[ c_i = [m_i + k \mod 26] \]
- $\text{Dec}_k(c_1 \ldots c_t): \text{output } m_1 \ldots m_t, \text{ where }$
  \[ m_i = [c_i - k \mod 26] \]
Is the Shift Cipher secure?

<table>
<thead>
<tr>
<th>Brute-force Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ No – only <strong>26</strong> possible keys!</td>
</tr>
<tr>
<td>▶ Given a ciphertext, try decrypting with every possible key</td>
</tr>
<tr>
<td>▶ Only one possibility will “make sense”</td>
</tr>
<tr>
<td>▶ Example of a <strong>brute-force</strong> or <strong>exhaustive-search</strong> attack</td>
</tr>
</tbody>
</table>
Brute-force Attack on Shift Cipher

Example

- Ciphertext uryybjbeyq
- Try every possible key and decrypt:
  - message under key 1 is: tqxxaiadxp
  - message under key 2 is: spwwzhzcwo
  - ...
  - message under key i is: helloworld
  - ...
Byte-wise Shift Cipher

- Alphabet of \texttt{bytes} rather than (English, lowercase) letters
- Works natively for arbitrary data!
- Use \texttt{XOR} instead of modular addition
- Essential properties still hold
# Hexadecimal (base 16) Notation

<table>
<thead>
<tr>
<th>Hex</th>
<th>Bits</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hex</th>
<th>Bits</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>
Hexadecimal (base 16) Notation

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

0xAF
► 0xAF = 16*A + F = 16*10 + 15 = 175
► 0xAF = 1010 1111
ASCII

- American Standard Code for Information Interchange
- Character encoding standard
- Byte-wise Shift Cipher: encode characters in ASCII
- 1 byte = 1 character = 2 hex digits
- Encoded using the ASCII table
### ASCII table

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Null</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Start of heading</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Start of text</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>End of text</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>End of transmit</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Enquiry</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Acknowledge</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Audible bell</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Backspace</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Horizontal tab</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Line feed</td>
<td>42</td>
<td>2A</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Vertical tab</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Form feed</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Carriage return</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift out</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Shift in</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Data link escape</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 1</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 2</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 3</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 4</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Neg. acknowledge</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Synchronous idle</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>End trans. block</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Cancel</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of medium</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Substitution</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Escape</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>File separator</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Group separator</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Record separator</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Unit separator</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>`</td>
</tr>
</tbody>
</table>

[https://hubpages.com/technology/What-Are-ASCII-Codes](https://hubpages.com/technology/What-Are-ASCII-Codes)
Useful observations

- Only **128** valid ASCII chars (**128** bytes invalid)
- Only 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, Formally

- $\mathcal{M} =$ strings of bytes
- $\text{Gen:}$ choose uniform $k \in \mathcal{K} = \{0x00 \ldots 0xFF\}$ i.e. there are 256 possible keys
- $\text{Enc}_k(m_1 \ldots m_t):$ output $c_1 \ldots c_t$, where
  \[ c_i = m_i \oplus k \]
- $\text{Dec}_k(c_1 \ldots c_t):$ output $m_1 \ldots m_t$, where
  \[ m_i = c_i \oplus k \]
Is this scheme secure?

- No – only 256 possible keys!
- Given a ciphertext, try decrypting with every possible key
- If ciphertext is long enough, only one plaintext will ”make sense”
- Can further optimize
  - First nibble of plaintext likely 0x4, 0x5, 0x6, 0x7 (assuming letters only)
  - Recover 2 key bits and reduce exhaustive search by a factor of 4.
Sufficient key space principle

Crypto Design Lesson One

- The key space must be large enough to make brute-force attacks impractical

Spoiler: necessary, but not sufficient
End

Reference: Section 1.3 of the book