CSCI 1800 Cybersecurity and International Relations

Secure Communication and Authentication

John E. Savage
Brown University
Outline

- Symmetric Cryptography
- Public-Key Cryptography
- Cryptographic Hash Functions
- Digital Signatures
- Diffie-Hellman Key Exchange
The Cryptographic Problem

• Goal: Alice needs to communicate securely with Bob, but Eve listens or interferes with conversation.
• Approach: Alice and Bob encrypt messages (they create ciphertexts) to keep them secure from Eve.
• Eve engages in cryptanalysis, tries to break cipher.
• Security by obscurity is dangerous. Once obscure method is discovered, all secrets are lost.
• Better to assume encryption method is known but that keys remain secret. Keys can be changed.
## Three Types of Notation

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
</tr>
<tr>
<td>D</td>
<td>1101</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000 000</td>
<td>0 0</td>
</tr>
<tr>
<td>1</td>
<td>000 001</td>
<td>0 1</td>
</tr>
<tr>
<td>2</td>
<td>000 010</td>
<td>0 2</td>
</tr>
<tr>
<td>3</td>
<td>000 011</td>
<td>0 3</td>
</tr>
<tr>
<td>4</td>
<td>000 100</td>
<td>0 4</td>
</tr>
<tr>
<td>5</td>
<td>000 101</td>
<td>0 5</td>
</tr>
<tr>
<td>6</td>
<td>000 110</td>
<td>0 6</td>
</tr>
<tr>
<td>7</td>
<td>000 111</td>
<td>0 7</td>
</tr>
<tr>
<td>8</td>
<td>001 000</td>
<td>1 0</td>
</tr>
<tr>
<td>9</td>
<td>001 001</td>
<td>1 1</td>
</tr>
<tr>
<td>10</td>
<td>001 010</td>
<td>1 2</td>
</tr>
<tr>
<td>11</td>
<td>001 011</td>
<td>1 3</td>
</tr>
<tr>
<td>12</td>
<td>001 100</td>
<td>1 4</td>
</tr>
<tr>
<td>13</td>
<td>001 101</td>
<td>1 5</td>
</tr>
<tr>
<td>14</td>
<td>001 110</td>
<td>1 6</td>
</tr>
<tr>
<td>15</td>
<td>001 111</td>
<td>1 7</td>
</tr>
<tr>
<td>16</td>
<td>010 000</td>
<td>2 0</td>
</tr>
</tbody>
</table>
# American Standard Code for Information Interchange (ASCII)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>&amp; #32</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>&amp; #64</td>
<td>Ø</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>&amp; #33 ;</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>&amp; #65 ;</td>
<td>À</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>&amp; #34 ;</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>&amp; #66</td>
<td>Ê</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>&amp; #35 ;</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>&amp; #67 ;</td>
<td>Ç</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>&amp; #36 ;</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>&amp; #68 ;</td>
<td>Ð</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>&amp; #37 ;</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>&amp; #69</td>
<td>Ê</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>&amp; #38 ;</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>&amp; #70</td>
<td>Ô</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>&amp; #39 ;</td>
<td></td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>&amp; #71</td>
<td>Ñ</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>&amp; #40 ;</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>&amp; #72</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>&amp; #41</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>&amp; #73</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>&amp; #42</td>
<td>/</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>&amp; #74</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>&amp; #43</td>
<td>:</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>&amp; #75</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>&amp; #44</td>
<td>;</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>&amp; #76</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>&amp; #45</td>
<td></td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>&amp; #77</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>&amp; #46</td>
<td></td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>&amp; #78</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>&amp; #47</td>
<td></td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>&amp; #79</td>
<td>Ò</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>&amp; #48</td>
<td></td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>&amp; #80</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>&amp; #49</td>
<td></td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>&amp; #81</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>&amp; #50</td>
<td></td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>&amp; #82</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>&amp; #51</td>
<td></td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>&amp; #83</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>&amp; #52</td>
<td></td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>&amp; #84</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>&amp; #53</td>
<td></td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>&amp; #85</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>&amp; #54</td>
<td></td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>&amp; #86</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>&amp; #55</td>
<td></td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>&amp; #87</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>&amp; #56</td>
<td></td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>&amp; #88</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>&amp; #57</td>
<td></td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>&amp; #89</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>&amp; #58</td>
<td></td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>&amp; #90</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>&amp; #59</td>
<td></td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>&amp; #91</td>
<td>À</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>&amp; #60</td>
<td></td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>&amp; #92</td>
<td>Å</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>&amp; #61</td>
<td></td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>&amp; #93</td>
<td>Æ</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>&amp; #62</td>
<td></td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>&amp; #94</td>
<td>Æ</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>&amp; #63</td>
<td></td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>&amp; #95</td>
<td>Æ</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com

ASCII notation is an octal triplet.
Mapping from Octal to Binary

- (space) = 040_{oct} = 000 100 000
- f = 146_{oct} = 001 100 110
- m = 155_{oct} = 001 101 101
- n = 156_{oct} = 001 101 110
- o = 157_{oct} = 001 101 111
- r = 162_{oct} = 001 110 010
- s = 163_{oct} = 001 110 011
- u = 165_{oct} = 001 110 101
Message Fragment in Binary

- Map message: no mon no fun to ASCII
- n ➞ 156  o ➞ 157  (space) ➞ 040
- m ➞ 155  o ➞ 157  n ➞ 156  (space) ➞ 040
- n ➞ 156  o ➞ 157  (space) ➞ 040
- f ➞ 146  u ➞ 165  n ➞ 156
- Concatenate bits to form integer message M = 0011011...
Symmetric Cryptography

- They agree on a common encryption method.
- Both Alice and Bob have the same secret key.
- Convert a text message to an integer M.
  - Example: no mon no fun
  - \156\157\040\155\157\156\040\156\157\040\146\165\156
  - Slashes between octal triplets are for humans only
  - M = 001 101 110 001 101 111 000 100 111 ...
- Encrypt M as C = E_K(M) using function E and key K.
- Decrypt C same way, M = E_K(C). K is secret. Symmetric!
Eve Attempts to Get Secret Key

• Ciphertext-only attack (least info)
  – Eve only has ciphertext.

• Known-plaintext attack
  – Eve is given plaintext-ciphertext pair(s).

• Chosen-plaintext attack
  – Eve chooses plaintext(s), gets ciphertext(s).
    She may choose plaintexts adaptively.

• Chosen-ciphertext attack (most info)
  – Eve chooses ciphertext, gets plaintext.
Ciphers Introduced in Today’s Lecture

- Substitution ciphers
- Polygraphic substitution ciphers
- One-time pads
- Binary one-time pads
- Advanced encryption standard (AES)
- Public-key cryptography (RSA)
- Digital signatures and hash functions
Substitution Ciphers

• **Substitution ciphers** permute letters in alphabet
  – E.g. Caesar replaced a letter by one three places away in the Latin alphabet.
  – Caesar(3): \(a\ b\ c\ d\ \ldots\ \ x\ y\ z\) is replaced by \(d\ e\ f\ g\ \ldots\ a\ b\ c\)

• **General substitution cipher** maps letters in an alphabet to a **fixed permutation** of the alphabet.

• Substitution ciphers are easily broken
  – Despite \(26!\) (\(~4\times10^{26}\) letter permutations, frequency of use of each letter in English text is stable.
  – Since **e** occurs **12.22%** of the time, easy to find the substitute for **e**. Letter **a** occurs **8.05%** of time.
Frequency of Letters in English
Polygraphic Substitution Ciphers

- Substitution ciphers may be applied to fixed-sized blocks (of two or three letters, say).
- **Substitution box** is 2D matrix that translates pairs.
  - \( E(01,10) = 0101 \)
  - \( E(11,10) = 1001 \)

- However, statistics also work here.
- But useful when combined with other methods.
Vigenère Cipher

- **Vigenère cipher** (1586) is a polygraphic cipher on blocks of \( m \) letters. Given letters \((l_1, l_2, \ldots, l_m)\), \( l_j \) is shifted cyclically by \( k_j \) places for \( 0 \leq k_j \leq 25 \).

  - E.g. If \( m = 3 \), \( k_1 = 2 \), \( k_2 = 1 \), \( k_3 = 3 \), \((a,g,z)\) mapped to \((c,h,c)\). Let’s encrypt **attackatdawn**

  - \((a,t,t)(a,c,k)(a,t,d)(a,w,n)\) ➞ \((c,u,w)(c,d,n)(c,u,g)(c,x,p)\)
  
  - Encrypted message is **cuwcdncugcqp**
  
  - If \( m \) is reasonably small, easily broken by statistics.
Vigenère Cipher

• If m is reasonably small, the Vigenère cipher is easily broken by statistics.
  – How would you do that?

• The integers can be derived from a text string
  – thequickbrownfoxjumpsoverthelazydog
  – Start alphabet at 0; a ↔ 0, b ↔ 1, ..., t ↔ 19, ..., z ↔ 25,
  – 19 7 4 16 20 8 2 10 1 17 14 22 13 5 14 23 9 20 12 15 18 14 21 4 17 19 7 11 0 3 14 6
  – Does this look like a random string?
    • How many times are digits repeated?
One-Time Pad

• **One-time pad** (Miller 1882) uses m random integers \( \{k_j | 1 \leq j \leq m\}, 0 \leq k_j \leq 25 \), to shift letters in a string of length \( \leq m \).
  
  – The \( j^{th} \) letter is shifted by \( k_j \) positions.

  • A real one-time pad might have **edible pages** of digits.

  – Both sender and receiver need to know shifts

  – Provides perfect security when \( m \geq \) message length

  – Fails when pad reused or string is longer than \( m \).

  • One-time pad encryption broken during Cold War.
Example of **Binary One-Time Pad**

- Let $M = \text{message bits}$, $K = \text{random bit seq.}$, $X = \text{cipher}$
- Cipher $X = E_K(M) = K \oplus M$ where $\oplus$ is vector XOR

- Example – $K$ is vector XORed with $M$ to produce $X$
  - $K = 100101001110011000101100010$
  - $M = 001101111001101111000100000$
  - $X = 101000110111110111101000010$

- If a key bit is 1, flip message bit. Otherwise nothing

- **Decrypt:**
  - XOR the key $K$ to the received message $X$
  - This reverses the bit flips
Binary One-Time Pad Again

• Message represented as n-bit binary string.
  – E.g. \( M = 010011 \) (a vector)
• Generate random n-bit string \( K \) (the key or one-time pad)
  – E.g. \( K = 100110 \) (a vector)
• XOR (⊕) is defined as \( 1 \oplus 0 = 0 \oplus 1 = 1 \) and \( 0 \oplus 0 = 1 \oplus 1 = 0 \)
• XOR message \( M \) with key \( K \) bit-by-bit to encrypt as \( X \).

\[
X = E_K(M) = M \oplus K
\]
  – E.g. \( E_K(M) = (0 \oplus 1)(1 \oplus 0)(0 \oplus 0)(0 \oplus 1)(1 \oplus 1)(1 \oplus 0) = 110101 \)
• Decrypt by encrypting \( X \) with \( K \)

\[
E_K(X) = X \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M
\]
Reuse of One-Time Pad Dangerous

\[ \text{SEND CASH} \oplus \text{plaintext} = C_1 \]

\[ \text{emoticon} \oplus \text{plaintext} = C_2 \]
XORing Two Encrypted Images

\[ C_1 = K \oplus M_1 \quad C_2 = K \oplus M_2 \quad C_1 \oplus C_2 = M_1 \oplus M_2 \]
Pseudo-Random Number Generators

- It is expensive to produce true random nos.
- Pseudo-random number generators (PRNGs) generate numbers that “look” random.

• Encryption algorithms can be used as PRNGs.
  - Encrypt a fixed string and represent it in binary
  - E.g. $E(\text{attackatdawn}) = 0100110101001110110$
Advanced Encryption Standard (AES) (Rough Sketch)

- **AES** (circa 2001) is a symmetric cipher whose inputs and outputs are 128-bit blocks. It uses an encryption key $K$ of length 128, 192 or 256 bits, denoted AES-128, AES-192, AES-256.
Advanced Encryption Standard (AES)

• When $K$ has 128 bits, AES computes $X_0 = M \oplus K$ and then executes 10 rounds.
  – Each round does a substitution, permutation, mixing of results, and an XOR’ing step.
  – It is too complicated to explain here.

• AES is highly secure but can be attacked using the time spent computing – this is a side channel attack

• In 2010 AES-256 was considered highly secure.

• AES-192 and AES-256 approved for US Top Secret!
Modes of Block Cipher Operation

• AES is a block cypher. Message bits grouped into blocks $B_1, B_2, B_3 \ldots$ of fixed size & encrypted

• Several ways to use block ciphers, some bad!

• Electronic Codebook mode (ECB)
  
  – Encode $C_j = E_K(B_j)$, Decode $B_j = D_K(C_j)$
  
  – Simple, resilient, but
  
    if coding deterministic, can reveal patterns.
Modes of Block Cipher Operation

• **Cipher-Block Chaining Mode (CBC)**
  – $C_0$ is initialization block. $B_1$, $B_2$, $B_3$, ... are data blocks
  – Encode $C_1 = E_K(B_1 \oplus C_0)$, $C_2 = E_K(B_2 \oplus C_1)$, etc.
  – Decode $B_j = D_K(C_j \oplus C_{j-1})$

  – If $C_j$ lost, can decrypt all but $j$th and $(j+1)$st block.
Modes of Block Cipher Operation

- **Cipher Feedback Mode** (CFB)
  - $C_0$ is the initialization vector.
  - Encode $C_j = E_K(C_{j-1}) \oplus B_j$, Decode $B_j = E_K(C_{j-1}) \oplus C_j$
  - No decryption, only encryption $E_K(\ldots)$. Faster than CBC.

- **Output Feedback Mode** (OFB)
  - $V_0$ is initial vector, $V_1 = E_K(V_0)$, $V_2 = E_K(V_1)$, and $V_j = E_K(V_{j-1})$
  - Encode $C_j = V_j \oplus B_j$, Decode $B_j = V_j \oplus C_j$
  - Like one-time pad – pad is an encryption sequence.
  - Can tolerate loss of blocks and can use parallelism
Public-Key Cryptography

• Each party has public & private keys
  – Alice: $\text{Priv}_\text{Alice}$, $\text{Pub}_\text{Alice}$; Bob: $\text{Priv}_\text{Bob}$, $\text{Pub}_\text{Bob}$.

• Alice encrypts message $M$ for Bob with
  \[ X = E_K(M) \text{ where } K = \text{Pub}_\text{Bob}. \]

• Bob decrypts Alice’s encrypted message with
  \[ M = E_{K^*}(X) \text{ where } K^* = \text{Priv}_\text{Bob}. \]

• Decrypt using same algorithm $E$ with private key
Origin of Public-Key Cryptography

• James Ellis, Clifford Cocks, Malcolm Williamson, invented it at GCHQ (British intelligence agency) by 1973, made public in 1997
• Diffie and Hellman propose idea publicly in ’76.
• Rivest, Shamir and Adleman (RSA) gave first practical implementation in 1977.

* http://en.wikipedia.org/wiki/Public-key_cryptography
Symmetric vs Public Key Crypto

• **Symmetric key system** has one key per user pair
  – Thus, there are $n(n-1)/2$ (pairs) keys for $n$ users
  – If $n = 10^4$, that’s about $50\times10^6$ keys!

• In **public-key system**, $2n$ keys suffice.
  – Each party publishes one key, keeps other secret

• Symmetric key system faster than public key.
  – PK systems often used to create/exchange secret symmetric keys
RSA Public-Key System

• Modular arithmetic
  – add and multiply integers modulo n
  – result is the remainder after dividing by n.
  – E.g. \((3+4) \mod 5 = 2\), \((4*3) \mod 3 = 0\)

• Bob’s public key \(\text{Pub}_B\) is the integer pair \((e,n)\).

• Bob’s secret key is \(\text{Priv}_B = d\). \(n = pq\), two primes

• Require that \(e\), \(d\), and \(n\) satisfy
  \[ X^{de} \mod n = X \] for any integer \(X\) in \(\{0,1,2,...,n-1\}\),
RSA Public-Key System

• Alice **encrypts** $M$ for Bob as $C = M^e \mod n$
  
  – Recall $\text{Pub}_B = (e,n)$

• Bob **decrypts** $C$ by computing $C^d \mod n = M$. This follows because
  
  $$C^d \mod n = (M^e)^d \mod n = M^{de} \mod n = M$$

• Bob can also **encrypt** $M$ as $C = M^d \mod n$ and decrypt with $C^e \mod n$ because
  
  $$C^e \mod n = (M^d)^e \mod n = M^{de} \mod n = M$$
Security of RSA

- Security dependent on difficulty of finding \( d \) given \( e \) and \( n \).
- Security closely tied to factoring \( n \). So far integer factorization is considered very hard to do.
- A mathematical proof of security of RSA is a very important open problem.
Cryptographic Hash Functions

- A **cryptographic hash function** compresses a message M into fixed-length sequence $H(M)$. Mapping is one-way and collision-resistant.
  - A function is **one-way** if it is computationally difficult to find M given $H(M)$.
  - It is **weakly collision-resistant** if it is difficult to find a message $M'$ with $H(M') = H(M)$ given just $H(M)$.
  - It is **strongly collision-resistant** if it is difficult to find both M and $M'$ with $H(M') = H(M)$.
Merkle-Damgard Hash Construction

• Divide M into length-m blocks M₁, M₂, M₃, ..., Mₖ.
• For |X|=m, |Y|=n, let C(X,Y), |C(X,Y)|= n, be a cryptographic hash function.
• Build H(M) from C(X,Y) as follows:
  – Let v = initialization vector
  – d₁ = C(M₁,v), d₂ = C(M₂,d₁), d₃ = C(M₃, d₂), ..., dₖ = C(Mₖ, dₖ₋₁)
  – H(M) = dₖ
• If H(M) = H(N), then H(M|P) = H(N|P) where | is concatenation. Thus C(X,Y) must be collision resistant.
• Secure hash algorithms SHA-256 and SHA-512 are based on this. They have been standardized by NIST.
Digital Signatures

• A **digital signature** of a message is a way for an entity to prove that the sender sent it.

• $S_{\text{Alice}}(M)$ denotes Alice’s signature on message $M$.
  – Alice “signs” a message $M$ by sending $(M, S_{\text{Alice}}(M))$.

• Digital signatures should be
  – **Non-forgery**
    • It should be difficult for Eve to create $S_{\text{Alice}}(M)$ from $M$.
  – **Non-mutable**
    • It’s hard for Eve to use $S_{\text{Alice}}(M)$ to create $S_{\text{Alice}}(N)$ on message $N$.

• Such signatures provide **non-repudiation**.
RSA Signature Scheme

- Bob’s public key $\text{Pub}_B = (e,n)$ & private key $\text{Priv}_B = d$
- Bob can sign message $M$ by providing to Alice both $M$ and $S = M^d \text{ mod } n$, the encryption of $M$.
- Alice deciphers $S$ with $(e,n)$ and recovers $M$. Bob shows he is source of $M$ because $d$ is secret.
- **Scheme is flawed** — doesn’t have non-mutability
  - If Eve has $(M_1, S_1 = M_1^e \text{ mod } n)$ & $(M_2, S_2 = M_2^e \text{ mod } n)$ she can create the signature for $M_1M_2$, which is $S_1S_2$, because $S_{1,2} = (M_1^e M_2^e) \text{ mod } n = (M_1 M_2)^e \text{ mod } n = S_1S_2$
  - RSA signature scheme needs to be fixed!
Digital Signatures with Hash Functions

• Here is how to form secure signatures

• Encrypt the hash $H(M)$ of a message $M$.
  – To sign $M$, compute $H(M)$ & encrypt $H(M)$ to produce the signature $\sigma = E(H(M))$.
  – Alice sends $(M, \sigma)$ to Bob.

• It is computationally difficult to decrypt and compute inverse of hash of the message when good hash function & strong encryption used.

• RSA signatures are rescued!
Digital Signatures

• Provide **integrity** & **non-repudiation**
  – Bob receives $M$ and $\sigma$. He decrypts $\sigma$ to produce $H(M')$ where $M'$ may be different from $M$.
  – Separately Bob computes $H(M)$ using his knowledge of the hash function $H$.
  – If the two results don’t match, no integrity.
  – Only authentic source can encrypt $H(M)$. 
Diffie-Helman Key Exchange

• Symmetric encryption is much faster than public-key encryption.

• Diffie and Helman invented a technique that two parties can use to agree on a secret key

• Both parties can use this key for symmetric encryption.
Diffie-Helman Key Exchange

• B & A choose prime $p$ & primitive root $g \mod p$.
  – $g$ is primitive if for each $r$ integer in \{0,1,2,..., p-1\}, $r$ satisfies $r = g^k \mod p$ for some integer $k$.

• Alice’s secret is $a$ and Bob’s secret is $b$.
  – A sends $r = g^a \mod p$ to B.
  – B sends $s = g^b \mod p$.
  – A computes $s^a \mod p$.
  – B computes $r^b \mod p$.

• Let $Q = s^a \mod p = (g^b \mod p)^a = g^{ba} \mod p = g^{ab} \mod p = r^b \mod p$. The common secret is $Q$!
Security of Diffie-Hellman

• The values of $a$ and $b$ are secret.
  – Alice sends $r = g^a \mod p$ to B in the clear.
  – Bob sends $s = g^b \mod p$ to Alice in the clear.
• These transmissions reveal $a$ and $b$ IF it is possible to deduce $a$ from $r = g^a \mod p$ or $b$ from $s = g^b \mod p$.
• This is the *discrete logarithm* problem.
• No polynomial time algorithm is known for it.
Review

• Symmetric Cryptography
• Public-Key Cryptography
• Cryptographic Hash Functions
• Digital Signatures
• Diffie-Hellman Key Exchange
Clicker Questions

• Q: Symmetric encryption is faster then public key
  – A True
  – B False

• Public key encryption makes good signatures
  – C True
  – D False