Outline

- Homework Solutions
- Attacks How-To
  - Vigenere Cipher
  - Homebrew Crypto
- Password Cracking
- Homework Assignment
Homework 2

Too easy...
How could you simultaneously ensure privacy and authenticity using public key crypto?

Solution:
- Assume Alice --> Bob
- We have PubKey_A, PrivKey_A, PubKey_B, PrivKey_B
- Enc(Message,PubKey_B) - Ensures that only Bob can read the message
- Enc(Message,PrivKey_A) - Ensures that Alice sent the message
Homework Problem 1

- How could you simultaneously ensure privacy and authenticity using public key crypto?

- Answer:
  - Enc(Enc(Message, PrivKey_A), PubKey_B)
    - Bob receives this message, decrypts with his private key and then with Alice's public key.

- Why not Enc(Enc(Message, PubKey_B), PrivKey_A)?
  - Anyone can strip off the outer encryption using Alice's public key (And replace it with their own)
    - Consider using this technique to submit homework
    - A MITM attack could intercept someone else's homework, strip off their sig, resign it, and submit it as their own
Homework Problem 2
Find the security flaw in the following server/client authentication protocol:

1) Client sends Challenge C1 to server
2) Server proves it’s identity by replying Enc(C1, SS)
3) User verifies server by encrypting C1 and comparing with message
4) Server sends C2 to client
5) Client authenticates by replying Encrypt(C2, SS)
6) Server verifies user by encrypting C2 and comparing message
Homework Problem 2

- To defeat this protocol, a client can open a new session and present the challenge C2 to the server.
- The response will be the Encryption of C2.
- The client responds to the original challenge with this message.

- How can you plug this security hole?
  - User authenticates first
Attack #1: Man in the Middle SSL

Credit: SSLStrip by Moxie Marlinspike (http://www.thoughtcrime.org/software/sslstrip/)
Credit: YouTube Demo Video by bonniekwacha (http://youtu.be/Q1hnHbBb_bA)
Overview:
1. Trudy enables packet forwarding
2. Trudy creates IPTable rule to redirect packets to SSLSTRIP
3. Trudy identifies Victim Host IP and Gateway/Router IP
4. Trudy spoofs the Gateway IP and advertises to the Victim IP
5. Trudy starts SSLSTRIP
6. Bob generates SSL traffic (tries to use secure socket service)
7. Trudy reviews logs generated by SSLSTRIP
8. Trudy obtains user credentials (usernames, passwords, etc)
Attacking A Substitution Cipher

Taken from DEFCON 18:

"AUJKKUKHNTPYMMJKKHYTGKHHMESELM
PFKDUJUSGFPEPVAFROPRERHOBAGY
JRAHWHLFVKWYBLZGBQHZZVUWKHM
MWRLAERQPAEREORAVLARALSMLMW
RLAEROPRERLUQAKPUQAZTMBRXLF
TSRLMLKBAUJKKKBNSTUMBFOGLPMK
DVMDKORBJKMEPEKIRZGYKMUZPGTSSGL"

Given hint: this problem was created by GMARK, and he loves Vigenere ciphers
Attacking A Substitution Cipher

Frequency analysis reveals that four of the five most occurring letters spell out "MARK". Knowing that GMARK created this challenge, let's try to decipher using these letters.

Unfortunately, this doesn't work out. Using GMARK as a key in a Vigenere cipher just results in gibberish. Any other ideas?
Attacking A Substitution Cipher

What if each letter in GMARK was an offset for five different alphabets? What if it were Vigenere, we just had different keys for each letter?

```java
String alpha = "ABCD EFGH IJKL MNOP QRST UVWX YZ";

//alphabet offset by G...
String galpha = "HIJK LMNO PQRS TUVW XYZA BCDE FG";

//alphabet offset by M...
String malpha = "NOPQ RSTU VWXY ZABC DEFG HIJK LM";

//alphabet offset by A...
String aalpha = "BCDE FGHI JKLM NOPQ RSTU VWXY ZA";

//alphabet offset by R...
String ralpha = "STUV WXYZ ABCD EFGH IJKL MNOP QR";

//alphabet offset by K...
String kalpha = "LMNO PQRS TUVW XYZA BCDE FGHI JK";

int alphaCounter = 0;
String decodedString = "";
int index = 0;
for (int a = 0; a < str.length(); a++)
{
    if (alphaCounter == 0)
    {
        index = galpha.indexOf(str.charAt(a));
        decodedString += alpha.charAt(index);
        alphaCounter++;
    }
    else if (alphaCounter == 1)
    {
        index = malpha.indexOf(str.charAt(a));
        decodedString += alpha.charAt(index);
        alphaCounter++;
    }
    else if (alphaCounter == 2)
    {
        index = aalpha.indexOf(str.charAt(a));
        decodedString += alpha.charAt(index);
        alphaCounter++;
    }
    else if (alphaCounter == 3)
    {
        index = ralpha.indexOf(str.charAt(a));
        decodedString += alpha.charAt(index);
        alphaCounter++;
    }
    else
    {
        index = kalpha.indexOf(str.charAt(a));
        decodedString += alpha.charAt(index);
        alphaCounter = 0;
    }
}
System.out.println(decodedString);
```
Success!

Decoded:
"THIS NUMBER IS ALSO A DATE ON WHICH SOMEONE DIED WHO ONCE PLAYED PLATO OPPOSITE A LEGEND. FIND THE DATE THE LEGEND DIED AND SEND HIM A MESSAGE AT THIS DOMAIN AND YOU WILL HEAR FROM BEYOND THE GRAVE"
Problem 3 - Attacking Home Brew Crypto

Decrypt This:
\xc0\x92\x29\x63\x1a\x50\x2c\xbd\x1a\x61\xfa\x75\xeb\xa6\xca\x95\x85\xce\xf9\x39\xc4\x5f\x1b
\x42\x50\xe1\x4e\xf8\x5f\xae\xe6\x53\xb2\x0b\xb7\x93\x1e\x85\x44\x94\xdf\xb8\xbd\x3b\x22
\x1e\x05\x73\x22\xe8\x75\x5f\x69\x3d\x6d\x4c\x62\xb5\x1b\x5f\xe5\x5b\x51\x91\xa3\x6e\x5d\x2a
\x52\x6f\x5e\x3d\x20\x2b\x96\xa7\xfa\x3c\xbf\xeb\x79\x94\x5f\x85\x8d\x00\x60\x8c\x8c\x21\xb7\x2c
\xda\xb6\x62\x6c\x5c\x7f\xb8\xc4\x2b\x75\xf8\xe5\x5f\x08\x3b\x2d\xf4\x14\x62\xf5\xe2\x93
\x39\xf5\x25\x95\xbb\x08\x13\x5b\x11\xa4\x2d\x52\xf7\x80\x91\x5f\x6b\x1e\xf0\x1c\xe1\xc5\x39
\x3f\xdb\x2a\x65\xc4\x16\x5f\xce\x3e\xff\xea\x1c\x4d\xa6\x57\xda\x82\xd0\x29\x9c\x74\xb9\xc4
\x8e\xe5\x76\xa2\xc3\xe8\xa8\x8e\x33\x30\xe3\x4d\x3f\x09\x20\x8f\x08\xe3\x3b\x8f\x74\x17
\x5e\x16\x6f\x82\x2b\x62\x74\xff\x5c\x6a\xa4\x2d\x7b\x03\xc3\xc7\xb5\xeb\xc4\x48\x09\x84\x94
\x86\x93\xaf\xd7\xf7\xd6\xff\x45\x09\xe7\x7b\x07\xc5\x7f\xad\x9d\xa4\x3e\x50\x9a\x73\x05\x1e\xe8
\x07\xbf\x5e\x4d\xc5\xf8\x3a\x87\x83\x5b\x3a\x24\x1d\x74\x34\x81\x8f\xe8\xd4\x52\xcb\x3d\xe0
\x5f\x46\x2d\x7b\x6d\x32\x7f\x53\xe5\x5b\x35\x7d\x61\xc5\xe8\xf7\x5f\xe2\x5f\x3f\xe8\x5a
\x70\x1b\x74\x82\x8c\x3c\x80\xa2\x5c\x88\x1e\x23\x5d\x3e\xb7\x33\x5f\xa9\x8d\xe8\xc5\x56\x51
\xbc\x93\x9d\x4d\x8c\x5f\x22\x72\x9f\x6d\x4b\x18\x1b\x02\xa5\xe6\xc9\x76\xc5\x52\x4d\x42\x4e\xdf
\xe7\x71\x09\x14\x8f\x71\x19\x16\x2a\x2c\x7e\xa5\xd9\xdc\x88\x4c\x53\xc6\x76\x1a\x15\xa4
\x25\xc5\xff\x8a\xb2\x61\x3e\xc6\x4b\x2f\x17\x73\x7b\xa6\x3e\x3a\x35\x4a\x1c\xa4\x88\x6f\xe8
\x1e\x8e\xe8\x99\x54\x4b\xc1\xcc\x7f\x5d\x19\x1b\x60\x28\xc0\xc0\xe2\x70\x1d\x6c\x3c\x3c
\x1b\x60\xad\x09\x50\x93\xcd\x30\xa0\xc4\x5a\x09\xfa\x1f\x23\x2a\x63\x94\x65\x5b\x56\x09\xd5
\xc6\x6d\xb3\x35\xe
\x03\x07\x70\x2c\x28\x33\xe8\x23\xe6\xba\x83\x15\x2c\x74\x30\xd5\x1a\x9a\x73\x6c\x2c\x58\xc3
\x25\x9a\x5f\x8f\x3c\x81\x74\x1b\xce\xe9\x8a\x80\xf8\x2f\x99\x2b\x06\xeb\x29\x52\xb1\x70\x14
\x7d\xec\x82\x52\x83\x9d\xe4\xc9\xc3\xdc\x51\xa7\x3e\x5d\x1e\x98\x1e\x8f\x82\x1c\x61\x60\x74
\x8b\x60\x9d\x92\x1f\xa4\x8c\x4d\x15\xb2\x44
def decrypt(msg, key):
    ks = len(key)
    k2 = key
    if(ks%2==1):
        k2 += chr(10)
        ks=ks+1
    cpt = ""
    buff = ""
    done = 0
    cnt = 0
    while(done==0):
        i=0
        nk = ""
        if(len(msg)>=cnt+ks):
            buff = msg[cnt:cnt+ks]
        else:
            buff = msg[cnt:]
        tcnt = len(buff)
cnt = cnt + tcnt
        if tcnt % 2 == 1:
            tcnt = tcnt - 1
        for i in range(0,tcnt,2):
            c1 = chr(((ord(buff[i])- 31) * -15) ^ (3 * ord(k2[i]))) % 256
            c2 = chr(((ord(buff[i+1])+17)^ (7*ord(k2[i+1])))+27*ord(k2[i]))*27 % 256
            while c1 < 0:
                c1 += 256
            while c2 < 0:
                c2 += 256
            nk = nk + c1 + c2
            cpt = cpt + c1 + c2
            i = tcnt
        if tcnt != len(buff):
            c1 = chr((ord(buff[i])-ord(k2[i]))% 256)
            while c1 < 0:
                c1 += 256
            nk = nk + c1
            cpt = cpt + c1
        k2 = nk
        if cnt == len(msg):
            done = 1
    return cpt
Still need the key!!

- Hint 2: The key and plaintext are both made up of printable ASCII characters

<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>
<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>@</td>
<td>96</td>
<td>60</td>
<td>`</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>
<td>97</td>
<td>61</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>
<td>98</td>
<td>62</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>
<td>99</td>
<td>63</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>
<td>100</td>
<td>64</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>
<td>101</td>
<td>65</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>
<td>102</td>
<td>66</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>
<td>103</td>
<td>67</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>
<td>104</td>
<td>68</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>
<td>105</td>
<td>69</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>
<td>106</td>
<td>6A</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>
<td>107</td>
<td>6B</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>
<td>108</td>
<td>6C</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>
<td>109</td>
<td>6D</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>
<td>110</td>
<td>6E</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>
<td>111</td>
<td>6F</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>
<td>112</td>
<td>70</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>
<td>113</td>
<td>71</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>
<td>114</td>
<td>72</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>
<td>115</td>
<td>73</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>
<td>116</td>
<td>74</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>
<td>117</td>
<td>75</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>
<td>118</td>
<td>76</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>
<td>119</td>
<td>77</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>
<td>120</td>
<td>78</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>
<td>121</td>
<td>79</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>
<td>122</td>
<td>7A</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>
<td>123</td>
<td>7B</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>
<td>124</td>
<td>7C</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>
<td>125</td>
<td>7D</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>
<td>126</td>
<td>7E</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>
<td>127</td>
<td>7F</td>
<td>_</td>
</tr>
</tbody>
</table>
Can we brute force it?

- Hint 3: The key is between 1 and 20 characters long.
  - Brute forcing a 20 character password of letters, numbers, and symbols?
  - 95 printable ASCII chars (127-32)
  - $95^{20} = 3,584,859,224,085,422,343,574,104,404,449,462,890,625$
  - No way we can brute force this...
Be Smart!

- Hint 4: How can we turn the problem of $95^{20}$ into something more like $95 + 20 \times 95 + 10 \times 95 + ...$

- Look closely at the decryption algorithm... Why should you never implement your own home brew crypto?
Attack Exercise - Password Cracking

Required Tools:
- BackTrack Linux (http://www.backtrack-linux.org/)
- John the Ripper (http://www.openwall.com/john/)*
  *already installed in BackTrack Linux (/pentest/passwords/john)

Provided Files:
http://www.utdallas.edu/~dst071000/CSG/HW3/pwdlist

Questions:
- Obtain as many passwords as possible from the provided password file
- Determine which hash algorithm was used to hash the passwords
- Locate the name of the system library that is used for hashing
- Identify expired accounts
- Identify passwords that were changed yesterday (09/13/2011)
- BONUS: Write a script that generates or verifies salted passwords

Hints:
- JTR Options (http://www.openwall.com/john/doc/OPTIONS.shtml)
- JTR Examples (http://www.openwall.com/john/doc/EXAMPLES.shtml)
Attack Exercise - Password Format

Linux

/etc/passwd

/etc/shadow
username:password:last:min:max:warn:inactive:expire

password
$number$salt$hash

unshadow
unshadow /etc/passwd /etc/shadow > combined.lst
Attack Exercise - Password Cracking

Review

Wordlists (Dictionaries):
   /pentest/passwords/wordlists/darkc0de.lst   (12 out of 20)
   /pentest/passwords/john/password.lst           (11 out of 20)
   (14 out of 20)

Command (Modes):
   john -single /path/to/password/file
   john -wordfile:/path/to/wordlist /path/to/password/file
   john -incremental /path/to/password/file
   ...

Command (Results):
   john -show /path/to/password/file
   rm john.pot
Attack Exercise - Password Cracking

Review

Answer Key:

● Obtain as many passwords as possible from the provided password file
  ○ user1=123456, user2 = 12345, user3=123456789, user4=Password, user5=iloveyou, user6=princess, user7=rockyou, user8=1234567, user9=12345678, user10=abc123, user11=monkey, user12=Qwerty, user13=Love, user14=Secret, user15=Xes, user16=God, user17=c0deb0Ok, user18=CrYpToGrApHy, user19=l337sp34k, user20=Supercalifragilisticexpialidocious

● Determine which hash algorithm was used to hash the passwords
  ○ $1$ indicates the MD5 hashing algorithm was used

● Locate the name of the system library that is used for hashing
  ○ crypt library

● Identify expired accounts
  ○ user6 and user15 are disabled accounts (expire < today) (today=15232)

● Identify passwords that were changed yesterday (09/13/2011)
  ○ user3 and user13 passwords (days since 01/01/1970 = 15231)

● BONUS: Write script to generate or verify salted password hash
  ○ Congratulations you get a Gold Star!!!
Advanced Password Cracking

Future:
- SSD Accelerated Rainbow Tables
- GPU Frameworks (Nvidia CUDA, ATI Stream, OpenCL, et. al.)

Alternate Tools:
- Cain & Abel (http://www.oxid.it/cain.html)
- RainbowCrack (http://project-rainbowcrack.com/)
- Online Lookup (http://www.onlinehashcrack.com)
- Hashcat / oclHashcat (http://hashcat.net/hashcat/)

Mitigation:
- STRONG........passphrases
  - long, case sensitive, numerals, special characters, encoded
- Use Windows 7 or Vista (NT hash > LM hash)
- Use SHA family, Avoid MD5 family
- Never store in plaintext
- Never authenticate over plaintext
- Use multi-factor authentication (two+)
- Everything tastes better with a little.......
This Week's Homework

Download problems 1 & 2 from www.tinyurl.com/solveus

Problem 1: Find the key!

Problem 2: Decrypt the message and find the secret word!