Report Project 2

Course code: IX1500
Date: 2018-10-01

Maher Jabbar, maherj@kth.se
Simon Lagerqvist,

Task : (1) RSA cipher

Summery

Task

Professor Alice is sending a message to the student Bob according to the procedure in section XXX.XXX.4. You are supposed to crack the message. When translating to ASCII, you can assume the base 256.

```
nBob = 126 456 119 090 476 383 371 855 906 671 054 993 650 778 797 793 018 127;
eBob = 7937;

cipher = {60 833 078 379 832 053 733 665 235 104 517 667 174 744 887 177 103 669,
    47 845 135 110 330 425 759 238 983 903 835 414 959 478 333 031 403 660,
    29 226 436 027 122 547 212 719 862 444 995 325 439 654 173 683 124 719,
    26 852 073 219 160 460 393 476 539 289 841 435 348 076 003 235 573 562,
    18 536 789 208 272 843 521 201 394 019 815 486 297 145 984 481 554 371,
    60 946 204 295 190 537 657 611 153 931 568 067 486 237 180 585 452 998,
    23 651 682 987 715 782 801 807 742 133 012 602 969 829 495 021 520 007,
    112 630 050 746 349 041 975 951 827 486 336 529 408 641 182 025 699 787,
    46 387 928 110 260 904 731 968 713 144 311 859 620 686 048 715 174 256,
    101 614 383 351 383 083 936 620 333 816 943 396 668 613 455 381 224 570};
```

- The numbers in the list seems to be of the same size. Why?

Result

1. The plain text is:
"Congratulations! You have now managed to crack the RSA cipher. This means that you have a pass grade for project 2. If you want to pursue a higher grade you need to solve one more problem."

2. The numbers in the list must be same size (and can not be less than nBob) otherwise the information would be lost while proceeding with Mod operation.

RSA cipher

The Model

The RSA decryption process that will be used in code section are in the following steps:

1. Extract the two prime numbers pBob and qBob from Bobs public key (nBob).
2. Finding the private key for Bob (dBob) by finding the modular inverse of eBob mod $(p-1)(q-1)$
3. Recovering plain text from cipher text by computing: $cipher^{dBob}\ mod\ (nBob)$

Code

In[•]:=
```
ClearAll["`*"]
nBob = 126 456 119 090 476 383 371 855 906 671 054 993 650 778 797 793 018 127;
(* public key *)
eBob = 7937; (* encryption key *)
base = 256; (* base is 256 *)
cipher = {60 833 078 379 832 053 733 665 235 104 517 667 174 744 887 177 103 669,
    47 845 135 110 330 425 759 238 983 903 835 414 959 478 333 031 403 660,
    29 226 436 027 122 547 212 719 862 444 995 325 439 654 173 683 124 719,
    26 852 073 219 160 460 393 476 539 289 841 435 348 076 003 235 573 562,
    18 536 789 208 272 843 521 201 394 019 815 486 297 145 984 481 554 371,
    60 946 204 295 190 537 657 611 153 931 568 067 486 237 180 585 452 998,
    23 651 682 987 715 782 801 807 742 133 012 602 969 829 495 021 520 007,
    112 630 050 746 349 041 975 951 827 486 336 529 408 641 182 025 699 787,
    46 387 928 110 260 904 731 968 713 144 311 859 620 686 048 715 174 256,
    101 614 383 351 383 083 936 620 333 816 943 396 668 613 455 381 224 570};
(* cipher text *)
(* extraction p and q from nBob *)
pAndq = FactorInteger[nBob];
pBob = pAndq[[1, 1]];
qBob = pAndq[[2, 1]];
(* finding decryption (private)key dBob *)
ϕBob = (pBob - 1) (qBob - 1);
dBob = PowerMod[eBob, -1, ϕBob];
(* cracking the cipher message *)
plaincode = PowerMod[cipher, dBob, nBob];
(* buffer to hold message *)
plaintext[0] := {}
plaintext[num_] := Prepend[plaintext[Quotient[num, base]], Mod[num, base]]
(* reconstruct the original message from ascii codes *)
FromCharacterCode[Join @@ plaintext /@ plaincode]
```

Out[•]=
```
Congratulations! You have now managed to crack the RSA cipher.
  This means that you have a pass grade for project 2. If you want
  to pursue a higher grade you need to solve one more problem.
```

Task : (2) RSA decryption analysis

Summery

Task

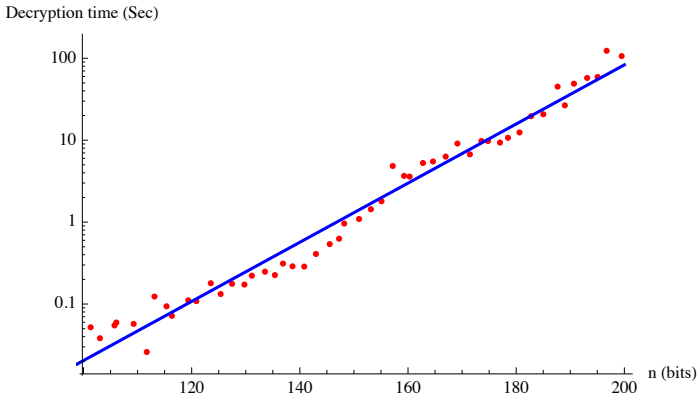
Write a method `RSAcrack[cipher,n,e]` that will crack a standard RSA cipher and delivers clear text from the string cipher. When you are finished with your method, you should investigate how long it will take to crack the cipher of the English text "ATTENTION THE UNIVERSE! BY KINGDOMS RIGHT WHEEL." for different sizes on your public key n (100-200 bits). Visualize your results in a proper graph. It is very important that you study the section 2.3 in the instructions. Your graph should lead you to a model where you can predict how long it would take to crack a cipher if n is 1024, 2048 bits or 4096 bits

- Motivate your selected mathematical model!

Result

1. The time for cracking the cipher of the English text "ATTENTION THE UNIVERSE! BY KINGDOMS RIGHT WHEEL." was **0.0876087** seconds

2. The decryption time vs key sizes (100-200 bits) from the output of code section (where the red points represents the normal data and the blue line represents the fitted data) is:



3. The prediction time (years) for different n (bits) is:

n (bits)	Decryption Time (years)
1024	1.45098×10^{24}
2048	1.32113×10^{61}
4096	1.09527×10^{135}

RSA decryption analysis

The Model

The comments in the code section provides information regarding the mathematical model that is used in encryption and decryption processes.

Code

```
In[•]:= ClearAll["`*"]

        (* Generate 2 random prime numbers (50-60) bits *)
        p = NextPrime[RandomInteger[{2^50, 2^60}]]
        q = NextPrime[RandomInteger[{2^50, 2^60}]]
```

```
Out[•]= 1 140 034 711 220 253 077
```

```
Out[•]= 407 911 063 164 465 181
```

```
In[•]:= (* Finding public and private keys (100 bits)*)
        n = p * q
        ϕ = (p - 1) (q - 1)
```

```
Out[•]= 465 032 771 098 247 474 895 212 504 174 611 937
```

```
Out[•]= 465 032 771 098 247 473 347 266 729 789 893 680
```

```
In[•]:= (* Generate random encryption key *)
        RandomSeed[];
        While[GCD[e = RandomInteger[{10^20, 10^30}], ϕ] ≠ 1];
        e
```

```
Out[•]= 246 887 640 620 478 078 740 700 934 997
```

```
In[•]:= (* Evaluate dncryption key *)
        d = PowerMod[e, -1, ϕ]
```

```
Out[•]= 117 879 920 915 019 989 858 578 074 078 996 173
```

```
In[•]:= (* the given plain text *)
        mes = "ATTENTION THE UNIVERSE! BY KINGDOMS RIGHT WHEEL.";
```

```
In[•]:= (* Breaking message into parts (each part is 8 bytes) *)
        mesLen = Range[StringLength[mes]] - 1;
        base = 256;
        buffer[0] := {}
        buffer[num_] := Prepend[buffer[Quotient[num, base^8]], Mod[num, base^8]]
        mesParts = buffer[ToCharacterCode[mes]. (base^mesLen)]
```

```
Out[•]= {5 713 190 297 443 521 601, 5 644 453 189 917 089 870, 2 315 207 907 610 023 497,
          4 919 987 194 625 546 562, 5 208 212 111 155 744 079, 3 336 117 587 843 424 340}
```

In[•]:=
```
(*** Encryption process ***)
cipher = PowerMod[mesParts, e, n]
```

Out[•]=
```
{367 506 590 861 571 440 673 576 877 361 672 089,
 201 150 355 700 743 870 243 854 501 960 849 767,
 24 167 944 698 878 847 510 754 999 550 831 335,
 227 587 766 067 852 566 367 591 399 428 658 979,
 254 037 900 587 904 333 909 734 513 203 968 462,
 197 084 203 568 602 584 852 089 262 584 532 786}
```

In[•]:=
```
(*** Decryption Method ***)

(* when ϕ is known *)
RSAonePart[cipher_, n_, e_, ϕ_] :=
 Module[{code, str},
  code = {};
  str = PowerMod[cipher, PowerMod[e, -1, ϕ], n];
  While[str ≠ 0,
   AppendTo[code, Mod[str, base]];
   str = Quotient[str, base]
  ];
  code;
  FromCharacterCode[code]
 ]

(* when ϕ is unknown) *)
RSAcrack[cipher_, n_, e_] := Module[{str, fact, ϕ, code},
  fact = FactorInteger[n];
  ϕ = Times @@ (fact[[All, 1]] - 1);
  code = {};
  StringJoin @@ (RSAonePart[#, n, e, ϕ] & /@ cipher)
 ]
```

In[•]:=
```
(*** Decryption Process ***)
AbsoluteTiming[RSAcrack[cipher, n, e]]
```

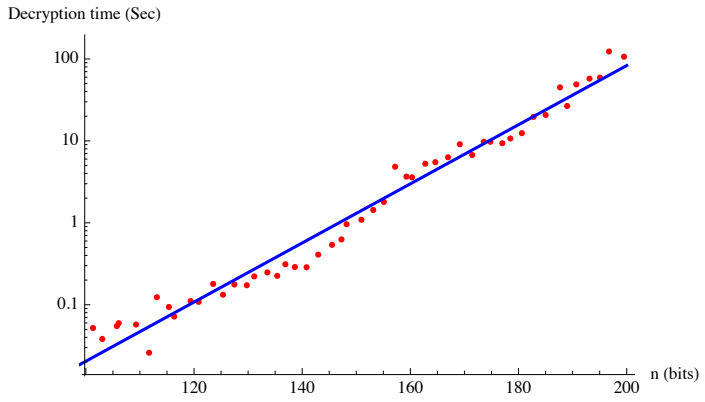
Out[•]=
```
{0.0876087, ATTENTION THE UNIVERSE! BY KINGDOMS RIGHT WHEEL.}
```

In[•]:=
```
(*** different sizes of public key (100-200 bits) ***)
time = {}; nBits = {};
RandomSeed[];
For[i = 0, i < 50, i++,
 p = NextPrime[RandomInteger[{2^(50 + i), 2^(51 + i)}]];
 q = NextPrime[RandomInteger[{2^(50 + i), 2^(51 + i)}]];
 n = p * q;
 ϕ = (p - 1) (q - 1);
 e = 0;
 While[GCD[e = RandomInteger[{10^20, 10^30}], ϕ] ≠ 1];
 cipher = PowerMod[mesParts, e, n];
 AppendTo[time, AbsoluteTiming[RSAcrack[cipher, n, e]][[1]]];
 AppendTo[nBits, N[Log[2, n]]];
]
```

In[•]:=
```
(*** Plotting time vs. key sizes ***)
fitData =
  FindFit[Transpose [{nBits, Log[time]}], slope * x + const, {slope, const}, x];
y[x_] = Exp[slope * x + const] /. fitData
listPlot = ListLogPlot[Transpose [{nBits, time}],
   AxesLabel → {"n (bits)", "Decryption time (Sec)"}, PlotStyle → Red];
fitPlot = LogPlot[y[x], {x, 50, 200},
   AxesLabel → {"n (bits)", "Decryption time (Sec)"}, PlotStyle → Blue];
Show[listPlot, fitPlot]
```

Out[•]= $e^{-12.201+0.0831073\,x}$

Out[•]=



In[]:=
```
(*** Calculating prediction time table ***)
t = 60 * 60 * 24 * 365; (* seconds to year *)
list = {{1024, y[1024] / t}, {2048, y[2048] / t}, {4096, y[4096] / t}};
TableForm[list,
 TableHeadings -> {None, {"n (bits)", "Decryption Time (years)"}},
 TableAlignments -> Center]
```

Out[]//TableForm=

n (bits)	Decryption Time (years)
1024	1.45098×10^{24}
2048	1.32113×10^{61}
4096	1.09527×10^{135}