18 April 2024

Once again, pull some previous things from the Crypto.mw file.

```
> with(StringTools):
```

Alphabet Setup

Let's define some default Alphabet by selecting all printable characters from the ASCII sequence. Probably you will change this depending on various things.

```
> #Alphabet := Select(IsPrintable, convert([seq(i,i=1..127)],
  bytes)):
  Alphabet:= cat(Select(IsAlpha, convert([seq(i,i=1..127)],
  bytes)), " ."):
  #Alphabet:= Select(IsUpper, convert([seq(i,i=1..127)], bytes)):
  #Alphabet:= Select(IsAlphaNumeric, convert([seq(i,i=1..127)],
  bytes)):
  printf("Our %d-character Alphabet is \n%s\n",length(Alphabet),
  Alphabet);

Our 54-character Alphabet is
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz .
```

StringToList, ListToString

StringToList converts a string into a list of numbers representing the position of each character in the Alphabet.
ListToString converts such a list back into a text string.

Note that this differ slightly from what we did in class, in that Alphabet[n] is represented by n-1. This will be more convenient when doing arithmetic.

```
> StringToList:=proc(str::string)
    global Alphabet;
    return(map( s->SearchText(s,Alphabet)-1, Explode(str)));
  end:
> ListToString:=proc(numlist::list(nonnegint))
    global Alphabet;
    return(Implode(map(k->Alphabet[k+1],numlist)) );
  end:
```

Vignère cipher

To encrypt, use as Vignere("plaintext", "secret key");
To decrypt, use as Vignere("crypttext", "secret key", decrypt); or
Vignere(crypttext, "decrypt key")

Adjusted from the earlier in-class to add the decrypt keyword. (and rewritten using a seq instead of a for loop)

```
> Vignere:= proc(msg::string, key::string, {decrypt::truefalse
  :=false})
    local numlist, alen, klen, i, shifted, shifts;
  global Alphabet;
  alen:=length(Alphabet); # length of alphabet
  if (decrypt) then
    return(Vignere(msg, ListToString(map(x->-x mod alen,
StringToList(key)))));
   fi;
  klen:=length(key); # length of key
  shifts:=StringToList(key); # turn my key into a list of
shifts
  numlist:=StringToList(msg); # convert message to numbers
  # now walk through the numlist, shifting by the appropriate
amount.
  shifted:=[seq(modp(numlist[i]+shifts[modp(i-1,klen)+1],
alen),i=1..length(msg))];
  return(ListToString(shifted));
end:
```

```
> Vignere("Once upon a midnight dreary while I pondered weak and
  weary", "Raven")
```

"fLVGlJNhPlrYfKOCGZJePBkGLGWtYSzJXcvPNhPOvPXFlLCTMlrLWchv ka" **(1)**

```
> Vignere(%,"Raven", decrypt);
```

"decrypting with ", "Raven"

"Once upon a midnight dreary while I pondered weak and weary" **(2)**

Note that even though this is harder to break than Caesar, you just need to figure out as many letters as the length of the key.

```
> Vignere("EEEEEEEEEEEEEK","Raven");
```

"VezirVezirVezix" **(3)**

BUT: A random key as long as the text is unbreakable... a one-time pad. This is because if we shift every character in the message by a different amount, any plaintext can correspond to any crypttext.

```
> message:="Once upon a midnight dreary while I pondered weak and
  weary"
```

message := "Once upon a midnight dreary while I pondered weak and weary" **(4)**

```
> length(message);
```

59 **(5)**

```
> key:=Random(59,Alphabet)
```

key := "FycUDYzMjvnE xwaTAriBM.PzSuyprRVfwjBbGxhXgS.Zn TXsbHjHlQURU" **(6)**

```
> Vignere(message,key);
```

"TjCyBQm UtLCkdXL.gWZ.pqtXHqwjWzEHur.OuiI.VwcXhctFq.uKFfuuGQ" **(7)**

```
> Vignere(%,key,decrypt);
```

"decrypting with ",

"FycUDYzMjvnE xwaTAriBM.PzSuyprRVfwjBbGxhXgS.Zn TXsbHjHlQURU"

"Once upon a midnight dreary while I pondered weak and weary" **(8)**

```
> key:=Random(59,Alphabet);
  Vignere(message,key);  # different key
  Vignere(%,key,decrypt);
```

key := "fhkCEiGmtHatnIMfarzhKqpGUjPydX NQY ehdXIo.garZxLrHpBHmcUKvb"

"tSKgCavYeF rXqpQGVeYIRekuYLwXCgyuWGcUPIlQqIBpTZlZFNokkWykkX"

"decrypting with ", "fhkCEiGmtHatnIMfarzhKqpGUjPydX NQY ehdXIo.garZxLrHpBHmcUKvb"

```
"Once upon a midnight dreary while I pondered weak and weary"                    (9)
> newkey:=Random(59,Alphabet)
newkey := "ghagLSIeTrjDPeQzUioMEZnnPgT.ZgrkhRmcItFGkLmPgBmcLnAULm ei.b"               (10)
```

The "Random" function is a pseudo-random process ... we can initialize it with our choice of "seed" to get the same random sequence again.

```
> Randomize(20240418);
                                 20240418                                           (11)
> key:=Random(59,Alphabet);
  newkey:=Random(59,Alphabet);
  newer:=Random(59,Alphabet);
key := "fQYyQgvwCmcNATASZSZwGzRyDYnpshCrnYPTkbiBWiAdynlsNG. hmFRbPX"
newkey := "O.lWsfbxlldDCbgeMyqjUFJaBVLMPXgEWmXfcXGFJuEEiNhFVPBADxDxDVN"
newer := "iIFYeEgwOO.YoOcKIQEzHDkHNUsoXL.MAvtyzkBWYx gaVuSv.jpH eVeFl"                 (12)
```

Reset the seed, we start the same sequence

```
> Randomize(20240418);   # reset the seed to choose the specific
  random sequence
  key:=Random(59,Alphabet);
  newkey:=Random(59,Alphabet);
  newer:=Random(59,Alphabet);
                                 20240418
key := "fQYyQgvwCmcNATASZSZwGzRyDYnpshCrnYPTkbiBWiAdynlsNG. hmFRbPX"
newkey := "O.lWsfbxlldDCbgeMyqjUFJaBVLMPXgEWmXfcXGFJuEEiNhFVPBADxDxDVN"
newer := "iIFYeEgwOO.YoOcKIQEzHDkHNUsoXL.MAvtyzkBWYx gaVuSv.jpH eVeFl"                 (13)
```

Let's write a fake one-time pad.

When I did this in class, I didn't realize that randomize insists on having an 8-digit or longer seed. So it failed sometimes.

```
> FakePad:=proc(message::string,seed::posint, {decrypt::truefalse
  :=false})
    global Alphabet;
    local cryptkey, randseed:=seed;
    if (randseed<87654321) then randseed:=seed+87654321; fi;  #
  hack to get a big seed
    randomize(randseed);  # where to start in the "one-time pad"
    cryptkey:=Random(length(message),Alphabet);
    print(cryptkey);
    if (decrypt) then
      return(Vignere(message,cryptkey,:-decrypt=true));
     fi;
    return(Vignere(message,cryptkey));
  end:

> cryptext:=FakePad("I think this worked.",314159);
  Alphabet;
                        "KaFqrbXuWVo.jmyiqmsN"
                   cryptext := "SYyVXMFsNAUrhgkXYOTM"
      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz ."                      (14)
```

```
> FakePad(cryptext,314159,decrypt);
                                        "KaFqrbXuWVo.jmyiqmsN"
                                           "I think this worked."                                 (15)
```

.

Discussion of affine cipher

want to be able to invert our encryption, so need factor to be relatively prime to base=length of Alphabet.

```
> 2*5 mod 6
                                                    4                                             (16)
> 3*5 mod 6
                                                    3                                             (17)
> seq(x*5 mod 6, x=0..5);
                                              0, 5, 4, 3, 2, 1                                    (18)
> seq(x*2 mod 6, x=0..5);
                                              0, 2, 4, 0, 2, 4                                    (19)
```

We hack up the Caesar cipher to make an Affine one.

```
> Affine:= proc(msg::string, m::integer, shift::integer,
       {decrypt::truefalse :=false})
    global Alphabet;
    local Alen:=length(Alphabet);
    local shifted;
    if (decrypt) then
       return("fail, try later");
     fi:
    shifted:= map(x-> modp(m*x+ shift,Alen), StringToList(msg));
    return(ListToString(shifted));
  end:
> Affine("ABC",2,1)
                                                  "BDF"                                           (20)
> Affine("ABC",6,0)
                                                  "AGM"                                           (21)
> length(Alphabet);
                                                   54                                             (22)
> gcd(15,3); gcd(15,4);
                                                    3
                                                    1                                             (23)

> Affine:= proc(msg::string, m::integer, shift::integer,
       {decrypt::truefalse :=false})
    global Alphabet;
    local Alen:=length(Alphabet);
    local shifted;
     if (gcd(m,Alen) <> 1) then
      error("m is not relatively prime to the length of the
  alphabet");
```

```
     fi;
    if (decrypt) then
        return(Affine(msg,modp(1/m,Alen),modp(-shift/m,Alen)));
     fi:
    shifted:= map(x-> modp(m*x+ shift,Alen), StringToList(msg));
    return(ListToString(shifted));
  end:
> Affine("ABC",2,0)
Error, (in Affine) m is not relatively prime to the length of the alphabet
> Affine("ABC",5,0)
                              "AFK"                              (24)
> Affine(%,5,0,decrypt);
                              "ABC"                              (25)
> Alphabet:= cat(Select(IsAlpha, convert([seq(i,i=1..127)], bytes)
  ), " "):
> length(Alphabet);
                               53                                (26)
> isprime(%);
                              true                               (27)
```

Still, this is easy to break, cuz 2 characters tell me everything.

```
> Affine("This is a secret",5,7);
                        "xNSPCSPCfCPzpKzU"                       (28)
```

IF I know that T->x and " "->C, I can break this.

```
> SearchText("T",Alphabet)-1; SearchText("x",Alphabet)-1;
                               19
                               49                                (29)
> SearchText(" ",Alphabet)-1; SearchText("C",Alphabet)-1
                               52
                                2                                (30)
> msolve({m*19+s=49, m*52+s=2}, length(Alphabet));
                          {m = 5, s = 7}                         (31)
```