April 25, 2024

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

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

$$Alphabet := \text{"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz '.,?"}$$

```
Our 67-character Alphabet is
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz '.,?
```

From last time.

```
> StringToList:=proc(str::string, {blocksize::posint:=1})
    global Alphabet;
    local Alen:=length(Alphabet);
    local numlist:=map( s->SearchText(s,Alphabet)-1, Explode(str)
  ) ;
    numlist := remove(x->x=-1, numlist);  # just kill any -1
  entries.
    if (length(str) > numelems(numlist)) then
      WARNING("%1 unrecognized characters omitted.",length(str)-
  numelems(numlist));
    fi;
    if (blocksize>1) then
      numlist:=convert(numlist,base,Alen,Alen^blocksize);
    fi;
    return(numlist);
  end:
```

```
> ListToString:=proc(nums::list(nonnegint),{blocksize::posint:=1}
  )
    global Alphabet;
    local Alen:=length(Alphabet);
    local numlist:=nums;
    if (blocksize>1) then
      numlist:=convert(numlist,base,Alen^blocksize,Alen);
    fi;
    return(Implode(map(k->Alphabet[k+1],numlist)) );
  end:
```

```
> Affine:= proc(msg::string, m::integer, b::integer:=0,
  {decrypt::truefalse :=false}, {blocksize::posint:=1})
    global Alphabet;
    local Alen:=length(Alphabet);
    if (gcd(m,Alen) <> 1) then
      error(sprintf("m=%d is not relatively prime to the length
  of the Alphabet=%d", m, Alen));
     fi;
    if (decrypt) then
      return(Affine(msg,modp(1/m,Alen^blocksize),modp(-b/m,
  Alen^blocksize),
          ':-blocksize'=blocksize));
     fi:
    local numlist:=StringToList(msg,':-blocksize'=blocksize);
    return(ListToString(map(x->modp(m*x+b,Alen^blocksize),
  numlist), ':-blocksize'=blocksize));
  end:
```

```
> Affine("Some Stuff",47);
```

"h5j4XhdJpp" **(1)**

```
> Affine(%,47,decrypt);
```

"Some Stuff" **(2)**

```
> Affine("Some Stuff",47,blocksize=5);
```

"hODczhwvND" **(3)**

```
> Affine(%,47,blocksize=5,decrypt);
```

"Some Stuff" **(4)**

One isssue is that trailing characters that encode to 0 get lost.

```
> Affine("This00000",2,blocksize=2);
  Affine(%,2,blocksize=2,decrypt);
```

"wJLg"

"This" **(5)**

```
> Affine("Give me a bunch of money.  How about 10000000000",2,
  blocksize=2);
  Affine(%,2,blocksize=2,decrypt)
```

"WLlEvUDw5w7kVAJwXGvUXWDszwvZXov67Yjiv3"

"Give me a bunch of money. How about 1" **(6)**

How to fix this? Later.

Salt makes things taste better.

```
> Affine:= proc(msg::string, m::integer, b::integer:=0,
  {decrypt::truefalse :=false}, {blocksize::posint:=1},
  {salts::nonnegint:=0})
    global Alphabet;
    local Alen:=length(Alphabet);
    if (gcd(m,Alen) <> 1) then
      error(sprintf("m=%d is not relatively prime to the length of
  the Alphabet=%d", m, Alen));
     fi;
    if (decrypt) then
      return(Affine(msg,modp(1/m,Alen^blocksize),modp(-b/m,
  Alen^blocksize),
```

```
          ':-blocksize'=blocksize));
     fi:
    local numlist:=StringToList(msg,':-blocksize'=blocksize);
    return(ListToString(map(x->modp(m*x+b,Alen^blocksize),
   numlist), ':-blocksize'=blocksize));
   end:
> Random(27,Alphabet);
                          "kl1H2XJC2?.KMOy,zDAbdnzXX7H"                    (7)
> Random(27,"Abc");
                          "AAAcAAAAbcAAcccbccAcbcAbcbb"                    (8)
```

idea: I want to add 1 noise char after each block of 2 text chars. blocksize=2, salts=1
"Hello!" -> "xHeKllo!"

```
> addsalt:=proc(str::string,salts,blocksize:=1)
    global Alphabet;
    local letters:=Explode(str);
    local saltylist:=[], grains;
    randomize();  # start the random number gen randomly.
    for local i from 1 to length(str) do
        if (blocksize=1 or modp(i,blocksize)=1) then
         grains:=Random(salts,Alphabet);
         else
          grains:="";
         fi;
        saltylist:=[op(saltylist), grains, letters[i]];
    od;
    return(cat(op(saltylist)));  # can't use implode if salts>1
   end:
> addsalt("this",2,1);
                              ".ZtH?hevib7s"                               (9)
> Affine:= proc(msg::string, m::integer, b::integer:=0,
   {decrypt::truefalse :=false}, {blocksize::posint:=1},
   {salts::nonnegint:=0})
    global Alphabet;
    local Alen:=length(Alphabet);
    if (gcd(m,Alen) <> 1) then
      error(sprintf("m=%d is not relatively prime to the length of
   the Alphabet=%d", m, Alen));
     fi;
    if (decrypt) then
       print("sorry, didn't do decryption");
      return(Affine(msg,modp(1/m,Alen^blocksize),modp(-b/m,
   Alen^blocksize),
          ':-blocksize'=blocksize));
     fi:
    local saltymsg:=addsalt(msg,salts,blocksize);
    local saltedsize:=salts+blocksize;
    local numlist:=StringToList(saltymsg,':-blocksize'=saltedsize);
    return(ListToString(map(x->modp(m*x+b,Alen^saltedsize),
   numlist),
         ':-blocksize'=saltedsize));
   end:
> addsalt("plain",2,2)
                             "JlplK2aiU7n"                                 (10)
```

```
> Affine("plain",1,salts=1,blocksize=2);
                                    "cplgai5n"                    (11)
> Affine("plain",5,salts=2,blocksize=3);  # so really this is
  encrypting 4 at a time (2 salt, 3 plain).
                                    "iZwbnS3Nl3"                  (12)
```

IT seems we are out of time I'll put this, and the decryption, into Crypto.mw