```
> with(StringTools):
> Alphabet:=cat("\n\t",Select(IsPrintable,convert([seq(i, i=1..255)
  ], bytes))):
  p:=length(Alphabet);
```

$$p := 97 \tag{1}$$

```
> StringToList := proc(text :: string)
    local i;
    global Alphabet;
    [seq(SearchText(text[i], Alphabet) - 1, i = 1 ..length(text) )];
  end:
  ListToString := proc(numlist :: list(nonnegint) )
    local i;
    global Alphabet;
    cat(seq(Alphabet[numlist[i] + 1], i = 1 ..nops(numlist) ) );
  end:

> StringTo2Vect:=proc( t::string)
    local L, s;
    s:=t;
    if ( modp(length(s),2) <> 0) then
        s:=cat(s,"X");
    fi;
    L:=StringToList(s);
    [seq(<L[i],L[i+1]>, i=1..nops(L)-1,2)];
  end:
  VectToString:=proc( vlist )
    ListToString(map(op,vlist))
  end:
> StringTo2Vect("blah blah blih");
```

$$\left[\begin{bmatrix} 68 \\ 78 \end{bmatrix}, \begin{bmatrix} 67 \\ 74 \end{bmatrix}, \begin{bmatrix} 2 \\ 68 \end{bmatrix}, \begin{bmatrix} 78 \\ 67 \end{bmatrix}, \begin{bmatrix} 74 \\ 2 \end{bmatrix}, \begin{bmatrix} 68 \\ 78 \end{bmatrix}, \begin{bmatrix} 75 \\ 74 \end{bmatrix}\right] \tag{2}$$

```
> VectToString(%);
```

Error, (in VectToString) invalid input: ListToString expects its 1st argument, numlist, to be of type list(nonnegint), but received [2, {1 = 68, 2 = 78}, datatype = anything, storage = rectangular, order = Fortran_order, shape = [], 2, {1 = 67, 2 = 74}, datatype = anything, storage = rectangular, order = Fortran_order, shape = [], 2, {1 = 2, 2 = 68}, datatype = anything, storage = rectangular, order = Fortran_order, shape = [], 2, {1 = 78, 2 = 67}, datatype = anything, storage = rectangular, order = Fortran_order, shape = [], 2, {1 = 74, 2 = 2}, datatype = anything, storage = rectangular, order = Fortran_order, shape = [], 2, {1 = 68, 2 = 78...

```
> op( <1,2>);
```

$$2, \{1 = 1, 2 = 2\}, datatype = anything, storage = rectangular, order = Fortran\_order, shape = [\,] \tag{3}$$

```
> convert( <1,2>, list);
```

$$[1, 2] \tag{4}$$

```
> op(%);
```

$$1, 2 \tag{5}$$

```
> VectToString:=proc( vlist )
    ListToString(map(x->op(convert(x,list)),vlist))
  end:
> VectToString(StringTo2Vect("yabba dabba do!"));
```

"yabba dabba do!X" (6)

```
> A:= < <1,0>|<2,4>>;
```

$$A := \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad (7)$$

```
> B:= convert([[1,2],[0,4]],Matrix);;
```

$$B := \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad (8)$$

```
> A.A;
```

$$\begin{bmatrix} 1 & 10 \\ 0 & 16 \end{bmatrix} \quad (9)$$

```
> A.A+ <<1,2>|<3,4>>;
```

$$\begin{bmatrix} 2 & 13 \\ 2 & 20 \end{bmatrix} \quad (10)$$

```
> vlist:=StringTo2Vect("Some Stuff");
```

$$vlist := \left[\begin{bmatrix} 53 \\ 81 \end{bmatrix}, \begin{bmatrix} 79 \\ 71 \end{bmatrix}, \begin{bmatrix} 2 \\ 53 \end{bmatrix}, \begin{bmatrix} 86 \\ 87 \end{bmatrix}, \begin{bmatrix} 72 \\ 72 \end{bmatrix} \right] \quad (11)$$

```
> wlist:=[seq( modp(A.vlist[i],p), i=1..nops(vlist))];
```

$$wlist := \left[\begin{bmatrix} 21 \\ 33 \end{bmatrix}, \begin{bmatrix} 27 \\ 90 \end{bmatrix}, \begin{bmatrix} 11 \\ 18 \end{bmatrix}, \begin{bmatrix} 66 \\ 57 \end{bmatrix}, \begin{bmatrix} 22 \\ 94 \end{bmatrix} \right] \quad (12)$$

```
> VectToString(wlist);
```

"3?9x)0`W4|" (13)

```
> modp(<123,456>,p);
```

$$\begin{bmatrix} 26 \\ 68 \end{bmatrix} \quad (14)$$

```
> Ainv:=modp(A^(-1),p);
```

$$Ainv := \begin{bmatrix} 1 & 48 \\ 0 & 73 \end{bmatrix} \quad (15)$$

```
> Ainv.A;
```

$$\begin{bmatrix} 1 & 194 \\ 0 & 292 \end{bmatrix} \quad (16)$$

```
> modp(%,97);
```

(17)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (17)$$

```
> cipro:=VectToString(wlist);
```

$$cipro := \text{"3?9x)0`W4|"} \quad (18)$$

```
> MatCrypt:= proc(text::string, A::Matrix)
    local v;
    v:=StringTo2Vect(text);
    VectToString([seq( modp(A.v[i],p), i=1..nops(v))]);
  end:
> MatCrypt("Some Stuff",A);
```

"3?9x)0`W4|" (19)

```
> MatCrypt(cipro,Ainv);
```

"Some Stuff" (20)

```
> MatDecrypt:=proc(crypt::string,B::Matrix)
    MatCrypt(crypt, B^(-1));
  end:

> MatDecrypt(cipro,A);
```

"Some Stuff" (21)

```
> Q:= <<0,1>|<1,0>>;
```

$$Q := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (22)$$

```
> MatCrypt("mix me up", <<0,1>|<1,0>>);
```

"im xemu Xp" (23)

want w->Aw+b

```
> AffineMatCrypt:= proc(text::string, A::Matrix, b::Vector)
    local v;
    v:=StringTo2Vect(text);
    VectToString([seq( modp(A.v[i]+b,p), i=1..nops(v))]);
  end:
> AffineMatCrypt("mix me up", <<0,1>|<1,0>>, <1,2>);
```

"jo!zfov"Yr" (24)

```
> MatCrypt("aaaabbbb",A);
```

"%h%h(l(l" (25)

```
> MatCrypt("abba",A);
```

"'l&h" (26)

```
> MatCrypt("aaaabbbb",<<1,0,0>| <1,2,3>|<4,5,6>>);
```

Error, (in LinearAlgebra:-Multiply) Vector dimension (2) must be the same as the Matrix column dimension (3)

```
> StringToVect:=proc( t::string, n::posint)
    local L, s;
    s:=t;
    while ( modp(length(s),n) <> 0) do
        s:=cat(s,"X");
    od;
    L:=StringToList(s);
    [seq(<seq(L[i+j], j=0..n-1)>, i=1..nops(L)-1,n)];
  end:
```

```
VectToString:=proc( vlist )
 ListToString(map(x->op(convert(x,list)),vlist))
end:
```

```
> StringToVect("short", 3);
```

$$\left[\left[\begin{array}{c}85\\74\\81\end{array}\right],\left[\begin{array}{c}84\\86\\58\end{array}\right]\right] \tag{27}$$

```
> StringToVect("short", 7);
```

$$\left[\left[\begin{array}{c}85\\74\\81\\84\\86\\58\\58\end{array}\right]\right] \tag{28}$$

```
> VectToString(%);
```

"shortXX" (29)

```
> with(LinearAlgebra):
  Dimension(A);
```

$$2, 2 \tag{30}$$

```
> AffineMatCrypt:= proc(text::string, A::Matrix, b::Vector)
   local v,n;
   n:=Dimension(A)[1];
   v:=StringToVect(text,n);
   VectToString([seq( modp(A.v[i]+b,p), i=1..nops(v))]);
  end:
> AffineMatDecrypt:= proc(text::string, A::Matrix, b::Vector)
   local v,n;
   n:=Dimension(A)[1];
   v:=StringToVect(text,n);
   VectToString([seq( modp(A^(-1).(v[i]-b),p), i=1..nops(v))]);
  end:
> AffineMatCrypt("stuff",<<1,0,0>| <1,2,3>|<4,5,6>>, <18,19,20>);
```

"RJ6$_" (31)

```
> AffineMatDecrypt(%,<<1,0,0>| <1,2,3>|<4,5,6>>, <18,19,20>);
```

"stuffX" (32)