QPP in a Nutshell

```
% message padded with K-1 zeros
plaintext = [ 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1
1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 ];
% security parameters
SP.blocksize = 4; % bits
% number of blocks
m = ceil(length(plaintext) / SP.blocksize);
% randon permutations
indices = sym(randi(factorial(2^SP.blocksize), 1, 8))
```

```
indices = (2678418387984  20903549180813  3580330116787  682100118137  11741865348054  18451107498
```

```
SP.permutations = [];
for i=indices
    SP.permutations = [SP.permutations; oneperm(2^SP.blocksize,i)-1 ];
end
disp( SP.permutations )
```

```
     2     0    12     3    11     1     7     4     9    14    15     5    13    10     8     6
    15    14    10    12     9    13     7     6    11     2     4     3     0     8     1     5
     2    12     0    15     8    11    14    13    10     6     7     4     9     1     3     5
     0     8    13     9     1     2    10     3    12     4     6     5    14    15     7    11
     8    15    10     9     2     6     4     1    11     0    13     7     3    14    12     5
    14     1    10     0    15     3     8    13     6     5    12    11     2     7     9     4
    10    11     8     5     7    15     2     0     4    13     9     1     6    14    12     3
     3     0    11    14     2     8     9     4     6    10     7    13    15     1    12     5
```

```
% encryption key
% maxi = 8;
% key = randi([1 maxi], 1, m);
key = [ 2     2     3     7     3     7     2     8     3     2     3
5     4 ];
% Encryption
ciphertext = encryption('QPP', SP, key, plaintext)
```

```
ciphertext = 1×50
     0     0     0     0     1     1     0     0     1     1     0     0     0···
```

```
% Decryption (for verification)
P = decryption('QPP', SP, key, ciphertext)
```

```
P = 1×50
     1     1     0     1     0     1     0     0     0     0     1     0     1···
```

```
% verify encryption and decryption
isequal( plaintext, P)
```

```
ans = logical
   1
```