QPP in a Nutshell

```
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 = (5036262727915  14146364902332  6048037297210  14056101090005  14544278614761  14225984(
```

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

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

```
% encryption key
maxi = 8;
key = randi([1 maxi], 1, m)
```

```
key = 1×13
     6     7     3     7     6     1     5     4     8     1     4     4     4
```

```
% Encryption
ciphertext = encryption('QPP', SP, key, plaintext)
```

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

```
% 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
```