Equivalent Key for pqsigRM

```
W1='/Users/dmoody/Desktop/pqsigrm3.txt'
f=open(W1)
L=f.readlines()
```

```
pk=L[6].rstrip()[5:]
```

```
def hextobin(hx):
    """
    return hex string to binary string
    """
    b=bin(ZZ(hx,base=16))
    return b[2:]

def hexbin(st):
    W=''
    for ct in range(0,len(st)):
        ts=hextobin(st[ct])
        while len(ts)<4:
            ts='0'+ts
        W=W+ts
    return W
```

```
PK=hexbin(pk)
```

```
len(PK)/4096
```

```
    1586
```

```
def dual(mat):
    mat1=mat.rref()
    T1=[]
    for j in range(0,mat1.nrows()):
        if mat1.row(j)==0:
            T1.append(j)
    mat2=mat1.delete_rows(T1)
    T3=[]
    T2=[]
    mr=mat2.rank()
    for j in range(0,mat2.ncols()):
        if j<mr and j not in mat2.pivots():
            T3.append(j)
        if j>mr-1 and j in mat2.pivots():
            T2.append(j)
    for j in range(0,len(T2)):
        mat2.swap_columns(T3[j],T2[j])
    mat2=mat2.rref()
    mat3=mat2.submatrix(0,mr,mr,mat2.ncols()-mr)
    mat4=mat3.transpose()
    i5=matrix.identity(GF(2),mat4.nrows())
    mat5=mat4.augment(i5)
    for j in range(0,len(T2)):
        mat5.swap_columns(T3[j],T2[j])
    return mat5
```

```
R=GF(2)
```

```
M=matrix(GF(2), 1586, 4096, lambda i, j: R(PK[j+4096*i]));
M
```

```
    1586 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
M.rank()
```

```
    1586
```

```
M2=dual(M)
M2
```

```
    2510 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
M2.rank()
```

```
    2510
```

```
M3=M.stack(M2)
M3
```

```
    4096 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
M3.rank()
```

```
    2570
```

```
M4=dual(M3)
M4
```

```
    1526 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
Z3=[]
for j in range(0,M4.ncols()):
    if M4.column(j)==0:
        Z3.append(j)
len(Z3)
```

```
    30
```

```
print Z3
```

```
    [154, 345, 571, 601, 958, 1123, 1430, 1471, 1739, 2021, 2186, 2195,
    2240, 2441, 2468, 2620, 2770, 2840, 2888, 2932, 2940, 3232, 3490,
    3535, 3591, 3875, 3936, 4004, 4067, 4084]
```

```
mr4=M4.rank()
mr4
```

```
    1526
```

```
flg=0
flg1=0
for c1 in range(0,mr4):
    for c2 in range(c1,mr4):
      r1=M4.row(c1)
      r2=M4.row(c2)
      if flg==0:
          M5=copy(M4)
          flg=1
      mm=matrix(GF(2),1,M4.ncols(),lambda i,j: r1[j]*r2[j])
      M5=M5.stack(mm)
      if c2==mr4-1:
          M5=M5.rref()
          W=[]
          for rw in range(0,M5.nrows()):
              if M5.row(rw)==0:
                  W.append(rw)
          M5=M5.delete_rows(W)
          mr=M5.rank()
          print c1,mr
          if mr==M5.ncols()-13-len(Z3):
              print 'done'
              flg1=2
              break
          if flg1==2:
              break
      if flg1==2:
          break
```

```
    0 2768
    1 3399
    2 3726
    3 3888
    4 3973
    5 4018
    6 4036
    7 4044
    8 4047
    9 4049
    10 4051
    11 4052
    12 4053
    done
    1525 4053
    done
```

```
M5
```

```
    4053 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
M6=dual(M5)
M6
```

```
    43 x 4096 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
M7=M6.rref()
M7
```

```
    43 x 4096 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
Msub=M7.submatrix(0,0,13,4096)
Msub
```

```
    13 x 4096 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
M8=matrix(Msub.row(0)+Msub.row(1))
for j in range(1,12):
    M8=M8.stack(matrix(Msub.row(j)+Msub.row(j+1)))
M8
```

```
    12 x 4096 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
ct=0
CSet=M8.columns()
CSet.sort()
for j in range(0,len(CSet)-1):
    if CSet[j]==CSet[j+1]:
        ct=ct+1
print ct
```

```
    30
```

```
ZM=matrix.zero(GF(2),1586,4096)
ZM
```

```
    1586 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
def coltobin(col):
    sm=0
    for j in range(0,12):
        sm=sm+2^j*ZZ(col[11-j])
    return sm
```

```
for j in range(0,4096):
    col=M8.column(j)
    tn=coltobin(col)
    ZM.set_column(tn,M.column(j))
```

```
ZM
```

```
    1586 x 4096 dense matrix over Finite Field of size 2 (use the
    '.str()' method to see the entries)
```

```
S.<x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12>=GF(2)[]
S
```

```
    Multivariate Polynomial Ring in x1, x2, x3, x4, x5, x6, x7, x8, x9,
    x10, x11, x12 over Finite Field of size 2
```

```
P=[]
for j in range(0,32):
    P.append(S.random_element(5))
```

```
def func(f1,j):
    st=bin(j)[2:].zfill(12)
    return
f1(ZZ(st[0]),ZZ(st[1]),ZZ(st[2]),ZZ(st[3]),ZZ(st[4]),ZZ(st[5]),ZZ(st[6]),ZZ(st[7]),ZZ(st[8]),ZZ(st[9]),ZZ(st[10]),
```

```
NM=matrix(GF(2), 1, 4096, lambda i, j: func(P[0],j));
```

```
for jj in range(1,32):
    NM1=matrix(GF(2), 1, 4096, lambda i, j: func(P[jj],j));
    NM=NM.stack(NM1)
NM
```

```
    32 x 4096 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
Z5=[]
for j in range(0,4096):
    if ZM.column(j)==0:
        Z5.append(j)
len(Z5)
```

```
    30
```

```
NM2=matrix(GF(2), 32, 30, lambda i, j: NM[i,Z5[j]]);
NM2
```

```
    32 x 30 dense matrix over Finite Field of size 2 (use the '.str()'
    method to see the entries)
```

```
B1=NM2.left_kernel().basis()
```

```
NM3=B1[0]*NM
```

```
NM3 in ZM.row_space()
```

```
    True
```