- View Shared Private Message Blockchain: using this module message owner or share users can decrypt and view all messages and non-sharing users cannot view or decrypt message.

5.2 SAMPLE CODE:

```
import bonaci

import Json

import base64

import base58

import nacl.hash

import nacl.signing

import nacl.secret

import nacl.utils

def to_b64(barray):

    return base64.b64encode(barray).decode('utf8')

def from_b64(string):

    return base64.b64decode(string)

def to_bytes(obj):

    if is instance (obj, bytes):

        return obj

    if isinstance(obj, dict):

        obj = json.dumps(obj, sort_keys=True, separators=(',', ':'))
```

```
    return obj.encode('utf8')

def sign(data, sign_key):

    return to_b64(sign_key.sign(to_bytes(data)))

 def random(size=nacl.secret.SecretBox.KEY_SIZE):

    return nacl.utils.random(nacl.secret.SecretBox.KEY_SIZE)

def hash(data):

    return nacl.hash.sha256(to_bytes(data)).decode('utf8')

def pkencrypt(data, sender_sk, receiver_pk):

    sender_sk = nacl.public.PrivateKey(base58.b58decode(sender_sk))

receiver_pk = nacl.public.PublicKey(base58.b58decode(receiver_pk))

     box = nacl.public.Box(sender_sk, receiver_pk)

    nonce = nacl.utils.random(nacl.public.Box.NONCE_SIZE)

    encrypted = box.encrypt(to_bytes(data), nonce)

    return to_b64(encrypted)

def pkdecrypt(data, sender_pk, receiver_sk):

sender_pk = nacl.public.PublicKey(base58.b58decode(sender_pk))

receiver_sk = nacl.public.PrivateKey(base58.b58decode(receiver_sk))

    box = nacl.public.Box(receiver_sk, sender_pk)

    return box.decrypt(from_b64(data))
```

```
def encrypt(data, key):
    box = nacl.secret.SecretBox(key)
    nonce = nacl.utils.random(nacl.secret.SecretBox.NONCE_SIZE)
    cipher = box.encrypt(to_bytes(data), nonce)
    return to_b64(cipher)
def decrypt(cipher, key):
    box = nacl.secret.SecretBox(key)
    decrypted = box.decrypt(cipher)
    return json.loads(decrypted.decode('utf8'))
def keypair(seed=None):
    if not seed:
        seed = nacl.utils.random(32)

signing_key = nacl.signing.SigningKey(seed=seed)
private_key = signing_key.to_curve25519_private_key()

    return {'sign': signing_key,
            'sign_b58': base58.b58encode(signing_key.encode()),
            'verify': signing_key.verify_key,
            'verify_b58': base58.b58encode(signing_key.verify_key.encode()),
            'private': private_key,
```

```
            'private_b58': base58.b58encode(private_key.encode()),
            'public': private_key.public_key,
            'public_b58': base58.b58encode(private_key.public_key.encode()),
            'seed': seed}
def create_keypair(name):
    filename = '.{}.bcdb_seed'.format(name)
    seed = nacl.utils.random(32)
    with open(filename, 'wb') as fh:
fh.write(seed)
def load_keypair(name):
    filename = '.{}.bcdb_seed'.format(name)
    with open(filename, 'rb') as fh:
        seed = fh.read()
    return keypair(seed)
def resolve(name):
    try:
        return load_keypair(name)['verify_b58']
    except FileNotFoundError:
        return name
```