Generated from BLS DKG (RUST)

Signature : (192 characters)

```
8286c632236bf789222f5866dcc441dbbdec4dd6ff76f468cf027648a47cd3f4e1be5de27fdf2e977284f9
97d97acf76187a93c6e85d98b16b78cec834d369bca04e7da41cb19ca40b686d4a5e52417de1932f4ec9c1
013c26e309488161f4eb
```

Public Key (384)

```
ac44cd8ae928f5524ccef9019eed2932bbf95d3726c3a9b17acc2edc5c645ad5335fb
912d2c5898a1d34e58114f748bd83789cea7d20cd19cc2d48618d16c30d2e94a5ed1e
e96494f70edd1ca3d7410774a860776acbc5792f403ba0007b116c85176f8720f1836
7069fdc504122db1987e1237b7b2da021d2c6f416bb09aa28683ac44b8a699cf28ac8
9b4a441709a6a6b53180d8ca0bb3f7666559a9a6e490df08a5d7177e019645a01b993
a4ee44ae10ec4a582a32b9ce207d3ca516c8abf
```

Generated from py_ecc Library (PYTHON)

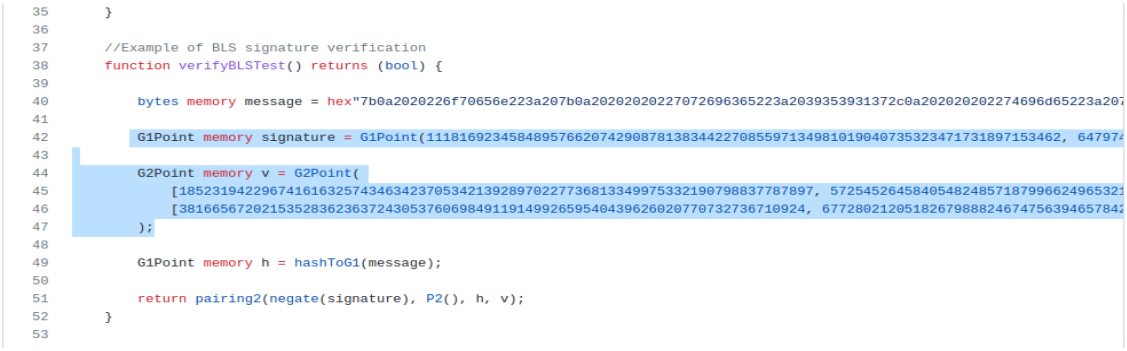
Public Key (96 characters)

```
98fad4ca8b98082de0ea80ba8e43ac4e7c45ad2730624b92f483370f8aaaf6514d58f
472012f7b7b37ae3a8b5af9ed13
```

Signature (192)

```
a95a0c6832bde37fc0b7bc9c14c46b487ad4f5dedf8866ac5b70200b9939e945903d0
dc34b78d7742832ee9234aebcd2105fdac4b33a23e1aa836879bbefbc669199223870
688ee10b3d545bad2549387c165c677c649a8eb4c0837336bee3d4
```

What is G1 and G2 ?



```
    }

    //Example of BLS signature verification
    function verifyBLSTest() returns (bool) {

        bytes memory message = hex"7b0a2020226f70656e223a207b0a20202020227072696365223a2039353931372c0a202020202274696d65223a207
        G1Point memory signature = G1Point(111816923458489576620742908781383442270855971349810190407353234717318971534 62, 647974
        G2Point memory v = G2Point(
            [185231942296741616325743463423705342139289702277368133499753321907988377878 97, 572545264584054824857187996624965321
            [381665672021535283623637243053760698491191499265954043962602077073273671092 4, 677280212051826798882467475639465784
        );

        G1Point memory h = hashToG1(message);

        return pairing2(negate(signature), P2(), h, v);
    }

```

Our smart contract takes public keys and signatures in G1 and G2 format.

Problem ?

1. Expected in smart contract : 77 characters
2. From Py_ECC Library : 115 characters

There is a difference in encoding that both python and bls dkg rust developers have used.

So the output for **G1** and **G2** of different sizes in **py_ecc** library.

Community Help 1

Issue : https://github.com/ethereum/py_ecc/issues/125

```
from py_ecc.bls import G2ProofOfPossession as bls_pop
from py_ecc.bls.g2_primitives import signature_to_G2, pubkey_to_G1

private_key = 5566
public_key = bls_pop.SkToPk(private_key)

message = b'\xab' * 32  # The message to be signed

# Signing
signature = bls_pop.Sign(private_key, message)

g1 = pubkey_to_G1(public_key)
g2 = signature_to_G2(signature)


print("-----------------------------------")
print(public_key.hex())
print("-----------------------------------")
print(signature.hex())
print("-----------------------------------")
print(bls_pop.Verify(public_key, message, signature))
```

Result :

Not worked, coz changing `G2ProofOfPossession` to `G2Basic/G2MessageAugmentation` does not impact in conversion public key to points.

References

1. Conversion of PK and SIG into G1 G2
https://github.com/ethereum/py_ecc/blob/master/py_ecc/bls/g2_primitives.py

2.BLS DKG RUST
https://github.com/maidsafe/bls_dkg/blob/master/src/key_gen/tests.rs

3. BLS Signature verifier Contract
https://gist.github.com/BjornvdLaan/ca6dd4e3993e1ef392f363ec27fe74c4#file-blsexample-sol