Carlos Augusto de Moraes Cruz
carlosamcruz@ufam.edu.br

# PreImage of BSV P2PKH Transaction of the type ForkID 41

Hexadecimal Transaction with a single input and two outputs:

01000000**01**a772376502aef2738dca0089811044c3689896ac37053f24a9784a13399f764a**01**0000008b4830450221**00**b8a72a38b9630139a597f271c7016501656d717012d14f81ceee47ad7bce8bf702203bc16d1098ed3b81d04e836b4ea6ed5791a6b0863c9165529215b7be3616ec1a**41**41044ff33350fbb662de8f22d488319792b563dbd120b016a1bc55645465ee27edcb7e58fbaa4bb7ff1cd2620882a18132a3e51e0b0da6455d0ff08703db5b6234b2**ffffffff**02e803000000000019**76a91461079f5031a5b7e312d9fc5051fd7ce018fabc9288acb01d000000000000**1976a9141f02307e6139effb4ec53283bcf6072e4796e10688ace9390b00

**Trasaction Parts:**

1) Version – 01000000

2) Number of Inputs – 01

3) TXID of the previous input (Big Endian - BE) –

a772376502aef2738dca0089811044c3689896ac37053f24a9784a13399f764a

4) Index of the previous TX being spent (Little Endian - LE) – 01000000

5) Number of bytes of input script (Signature DER + PubKey SEC) – 8b

**6) Signature Parts:**

4830450221**00**b8a72a38b9630139a597f271c7016501656d717012d14f81ceee47ad7bce8bf702203bc16d1098ed3b81d04e836b4ea6ed5791a6b0863c9165529215b7be3616ec1a41

6.1) 48 – size of signature (ECDSA) + integer number flags + format (DER) + type (ForkID 41);

6.2) 30 DER format;

6.3) 45 size of the ECDSA signature + integer number flag;

6.4) 02 integer flag of the r part of the ECDSA signature;

6.5) r part of the ECDSA signature:

2100b8a72a38b9630139a597f271c7016501656d717012d14f81ceee47ad7bce8bf7

6.6) 02 integer flag of the s part of the ECDSA signature;

6.7) s part of the ECDSA signature:

203bc16d1098ed3b81d04e836b4ea6ed5791a6b0863c9165529215b7be3616ec1a

6.8) 41 signature type ForkID 41;

**7) Public Key Parts:**

7.1) size of the SEC Format Public Key – 41

7.2) SEC Format Public Key

**044ff33350fbb662de8f22d488319792b563dbd120b016a1bc55645465ee27edcb7e58fbaa4bb7ff1cd2620882a18132a3e51e0b0da6455d0ff08703db5b6234b2**

8) Sequence of the Input (Same value for every input) - ffffffff

Carlos Augusto de Moraes Cruz
carlosamcruz@ufam.edu.br

9) Number of Outputs – 02

10) Satoshi amount paid to the 1st output (LE format) - e803000000000000

11) First Output Script:

1976a91461079f5031a5b7e312d9fc5051fd7ce018fabc9288ac

11.1) 19 Script size; 76 a9 14 OP_CODEs;

11.2) Address to pay (RIPMD160 hash of the Public Key):

61079f5031a5b7e312d9fc5051fd7ce018fabc92

11.3) 88 ac OP_CODEs;

12) Satoshi amount paid to the 2nd output (LE format) - b01d000000000000

13) Size and Script of the second output:

1976a9141f02307e6139effb4ec53283bcf6072e4796e10688ac

14) Locktime of the transaction - e9390b00

**PreImage Format**

1- Transaction version

2- Previous transaction outputs identifiers

3- Transaction input sequence numbers

4- The identifier of the output being spent

5- The locking script of the output being spent

6- The value of the output being spent

7- The sequence number of the transaction input

8- The created transaction outputs

9- Transaction locktime

10- The signature hash type

```
//https://www.reference.cash/protocol/blockchain/transaction/transaction-signing

////////////////////////////////////////////////////////////////////////////////
//Step 1: Transaction version (4-byte little endian)
////////////////////////////////////////////////////////////////////////////////

TXVersion = "01000000";

////////////////////////////////////////////////////////////////////////////////
//Step 2: Previous transaction outputs identifiers (32-byte hash)
////////////////////////////////////////////////////////////////////////////////

//previous outputs hash    32 bytes   hash(BE)
//A double SHA-256 hash of the set of previous outputs spent by the inputs of the transaction.
//Each TXID information the previous TX HASH must be in BE format

String prvOutHASH;

String PREVTXID = "a772376502aef2738dca0089811044c3689896ac37053f24a9784a13399f764a";

//output index 4 bytes   unsigned integer(LE)   The index of the output to be spent from the
specified transaction.
String pvOutIndex = "01000000";

prvOutHASH = PREVTXID + pvOutIndex;
```

Carlos Augusto de Moraes Cruz
carlosamcruz@ufam.edu.br

```
prvOutHASH = SHA256(SHA256(prvOutHASH));

////////////////////////////////////////////////////////////////////////////////
//Step 3: Transaction input sequence numbers (32-byte hash)
////////////////////////////////////////////////////////////////////////////////

//sequence numbers hash    32 bytes   hash(BE)
// A double SHA-256 hash of the set of sequence numbers of the inputs of the transaction.
// See Sequence Numbers for the hash preimage format.
//
//If hash type is “ANYONECANPAY”, “SINGLE”, or “NONE” then this is all 0x00 bytes.

String inSeq = "ffffffff";

String inSeqDHash;
inSeqDHash = SHA256 (SHA256(inSeq));

////////////////////////////////////////////////////////////////////////////////
//Step 4: The identifier of the output being spent (32-byte hash + 4-byte little endian)
////////////////////////////////////////////////////////////////////////////////

//previous output hash 32 bytes   hash(LE)   The transaction ID of the previous output being spent.
//previous output index    4 bytes    unsigned integer(LE)   The index of the output to be spent.
String prevOutID = PREVTXID + pvOutIndex;

////////////////////////////////////////////////////////////////////////////////
//Step 5: The locking script of the output being spent (serialized as scripts inside CTxOuts)
////////////////////////////////////////////////////////////////////////////////

//This information must be retrieved from transaction TXID being spent in the current input;

//modified locking script length   variable   variable length integer    The number of bytes for
modified_locking_script.
//Locking script  locking_script_length bytes   bytes(BE)
// The subset of the locking script used for signing.

String lockingScript = "19" + "76a914" + "1f02307e6139effb4ec53283bcf6072e4796e106" + "88ac";

////////////////////////////////////////////////////////////////////////////////
//Step 6:value of the output spent by this input (8-byte little endian)
////////////////////////////////////////////////////////////////////////////////

//previous output value    8 bytes    unsigned integer(LE)   The value of the transaction output
being spent.

String prevOutSatValue = "6022000000000000"; //Total satoshis from the previous output

////////////////////////////////////////////////////////////////////////////////
//Step 7: The sequence number of the transaction input (8-byte little endian)
////////////////////////////////////////////////////////////////////////////////

//input sequence number    4 bytes    unsigned integer(LE)   The sequence number of the input this
signature is for.

String inSeqNumber = "ffffffff";

////////////////////////////////////////////////////////////////////////////////
//Step 8: The created transaction outputs - hashOutputs (32-byte hash)
////////////////////////////////////////////////////////////////////////////////

//transaction outputs hash 32 bytes   hash(BE)   A double SHA-256 hash of the outputs of the
transaction.
// See Transaction Outputs for the hash preimage format.

//value    8 bytes    unsigned integer(LE)   The number of satoshis to be transferred.
//locking script length    variable   variable length integer    The size of the locking script in
bytes.
//locking script   variable   bytes(BE)  The contents of the locking script.

String TxOutputs = "e803000000000000" + "19" + "76a914" + "61079f5031a5b7e312d9fc5051fd7ce018fabc92"
+ "88ac"
        + "b01d000000000000" + "19" + "76a914" + "1f02307e6139effb4ec53283bcf6072e4796e106" + "88ac";

String TxOutputsDHASH = SHA256(SHA256(TxOutputs));

////////////////////////////////////////////////////////////////////////////////
//Step 9: nLocktime of the transaction (4-byte little endian)
////////////////////////////////////////////////////////////////////////////////

//transaction lock time    4 bytes    unsigned integer(LE)   The lock time of the transaction.

String nLockTime = "e9390b00";

////////////////////////////////////////////////////////////////////////////////
//Step 10: sighash type of the signature (4-byte little endian)
////////////////////////////////////////////////////////////////////////////////
```

Carlos Augusto de Moraes Cruz
carlosamcruz@ufam.edu.br

```
//hash type    4 bytes    Hash Type(LE)  Flags indicating the rules for how this signature was
generated.

String  sighashType = "41000000";

////////////////////////////////////////////////////////////////////////////////
//Transaction preimage
////////////////////////////////////////////////////////////////////////////////

String  TxPreimage = TXVersion + prvOutHASH + inSeqDHash + prevOutID + lockingScript
        + prevOutSatalue + inSeqNumber + TxOutputsDHASH + nLockTime + sighashType;
```

**Signature Validation:**

```
//TX signature in DER format
String signDER = "3045022100b8a72a38b9630139a597f271c7016501656d717012d14f81ceee47ad7bce8bf7" +
        "02203bc16d1098ed3b81d04e836b4ea6ed5791a6b0863c9165529215b7be3616ec1a41";

//Convert TX signature form DER format to BigInteger
BigInteger [] sigDERrev = pubKey.sigDERrev(signDER.substring(0,signDER.length()-2));

//TX Public Key in SEC format
String PubKeySEC = "41" + "044ff33350fbb662de8f22d488319792b563dbd120b016a1bc55645465ee27edcb" +
        "7e58fbaa4bb7ff1cd2620882a18132a3e51e0b0da6455d0ff08703db5b6234b2";

//Convert TX signature form SEC format to BigInteger
BigInteger [] pubKeySECUncREV = pubKey.pubKeyUncompSECRev(PubKeySEC.substring(2));


//ECDSA signature verifier

EcdsaSecretus ECDSA = new EcdsaSecretus();

//ECDSA VERIFY

//Double HASH of PreImage in BE format
String e = SHA256(SHA256(TxPreimage));


//ECDSA Validation
boolean signValidation = ECDSA.ECDSAVerifyBSV(e, pubKeySECUncREV, sigDERrev);
```