BIP 11 - LINK

M-of-N Standard Transactions - *(Multisig)*

What does BIP 11 do for Bitcoin

At the highest level BIP 11 is designed to enable secure wallets, escrow transactions, and other use cases where the exchange of funds **requires more than a single signature (key).**

What does M of N mean? Why was BIP 11 needed?

The term M of N means that you need to possess M number of keys (signatures) out of N number of accepted keys to spend a bitcoin transaction. An example of this would be a treasure chest with three keyholes and you would only need two keys to unlock the chest and retrieve the treasure inside.

BIP 11 was needed because of the fact that a few years after the release of Bitcoin. Transactions needed to pass through the **IsStandard()** function. This function is a filter that will allow a TX to propagate to the mempool and be relayed if the TX is a standard type. Defined in ‘src/script/standard.cpp’ At the time of the writing of BIP 11 - Multisig was not a standard transaction type and as such would not be relayed through the network. This led to the creation of P2MS which represents ~1% of all transactions in the UTXO set

Simply Put

M of N allows a transactions such as these examples to work.

2 of 3 example: You can build a transaction that has 3 input ‘*seed phrases*’ and the bitcoin is spendable if just 2 of them are presented.

2 of 2 example: This transaction has and would accept only 2 valid ‘*seed phrases*’ - Making the total transaction security equal to 512 bits.

- *If you have ever used server grade computer hardware this is similar to a RAID5 configuration where you have M number of disks and if N number of them fail your data is still preserved.*

USE CASES - N of N

N of N use cases

- Use of wallet protection services to provide keys
- Augmenting the security of a potentially compromised computer
- Augment RNG with a second source
- 2 Factor Authentication Spending

USE CASES - N of M

ESCROW - Confidence in a transaction can be increased by using a 2 of 3 MULTISIG where there is a buyer, seller, and a trusted dispute agent involved with a transaction.

This is achieved by having the three participants sign the 2 of 3 TX. Then the transaction can broadcast (not spent). Firstly the goods must be delivered to the buyer. After delivery the buyer then signs his portion of the TX and funds are released to the seller. If goods are delivered but the buyer is nefarious and refuses to hand over the keys. Then the seller will have the trusted dispute agent sign the transaction with their key allowing the seller to be made whole. The reverse can be true for the buyer.

Standard Transaction Types - Why do they matter?

BIP 11 was needed because of the fact that a few years after bitcoins release code was implemented such that Bitcoin transactions had to pass through the IsStandard() function. This function is a filter. It will allow a TX to propagate to the mempool and be relayed if the TX is of a defined a standard type.

These are listed in … ‘src/script/standard.cpp’

[SCRIPTHASH, WITNESS_V0_KEYHASH, WITNESS_V0_SCRIPTHASH, WITNESS_V1_TAPROOT, WITNESS_UNKNOWN, NULL_DATA, PUBKEY, PUBKEYHASH, MULTISIG] are all valid IsStandard() types.

Any bitcoin transaction script that doesn’t pass as one of these types will not get relayed through the network successfully. Any scripts that are mined into a block are executed.

Limits of (P2MS) Multisig Combinations:

Supports x of 3 per '*bitcoin/src/policy/policy.cpp*'

```
// Support up to x-of-3 multisig txns as standard
if (n < 1 || n > 3)
    return false;
if (m < 1 || m > n)
    return false;
```

P2SH (Pay to Script Hash) has largely replaced P2MS

Limitations

Because of the way OP_CHECKMULTISIG is counted by old clients as using 20 sigops and blocks are limited to 20,000 sigops. It is impossible to use more than 1000 multisig transactions per block. OP_CHECKSIG offers a way sound this limitation.

“A weaker argument is OP_CHECKMULTISIG should not be used because it pops one too many items off the stack during validation. Adding an extra OP_0 placeholder to the scriptSig adds only 1 byte to the transaction, and any alternative that avoids OP_CHECKMULTISIG adds at least several bytes of opcodes.”

LINKS

https://github.com/bitcoin/bips/blob/master/bip-0011.mediawiki

https://en.bitcoin.it/wiki/OP_CHECKMULTISIG

https://learnmeabitcoin.com/technical/p2ms

https://bitcoin.stackexchange.com/a/28092