How to broadcast a Bitcoin payment transaction via a single SMS

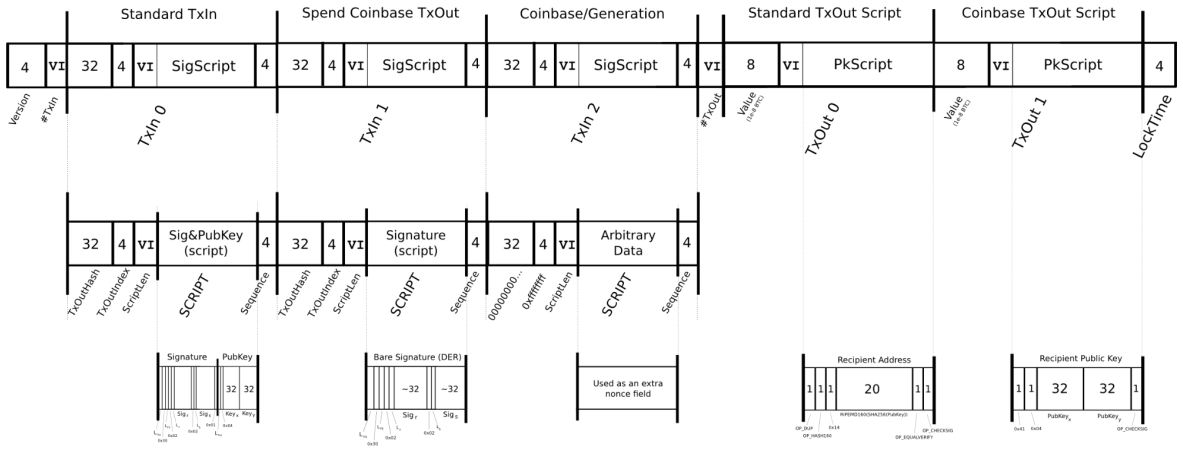
bohdan.skriabin@distributedlab.com
oleksandr.kurbatov@distributedlab.com

Abstract. We propose a method for sending Bitcoin payments when the sender has no Internet connection. However, the sender has a Bitcoin address to which coins need to be sent, and somewhere on the planet, there is an intermediary who can receive SMS, rebuild the full transaction, and propagate it across the network. The essence of the method lies in creating a transaction, signing, discarding unnecessary data, and sending it via SMS to the intermediary who knows how to reconstruct the discarded data to make the transaction valid and confirmed in Bitcoin.

1. Previous research

While exploring solution options, various approaches were considered, including using SIGHASH_SINGLE, SIGHASH_NONE, or SIGHASH_ANYONECANPAY. Additionally, the option of sending a private key via SMS was explored. However, these methods require more trust in the SMS transmission service and the intermediary. Hence, we will focus on the method with the omission of certain transaction data, which can be reconstructed by the intermediary as long as both the sender and intermediary adhere to the same transaction formation rules.

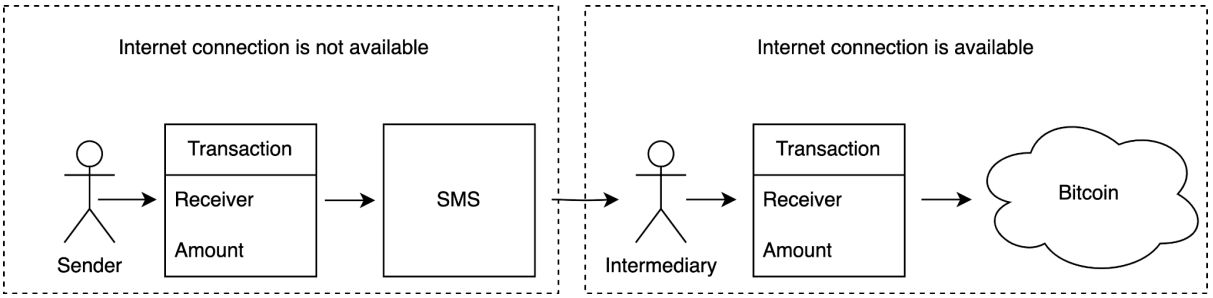
For more efficient compression, we will start from the very beginning by selecting the smallest transaction, namely a transaction with a single input. Typically, such transactions have a size of 192 bytes if they contain one output and 240 bytes if they contain two outputs (the second being the change output). But the usual SMS contains up to 140 bytes [1]. Converting bytes to symbols is simple we can use an Ascii85 (Base85) encoder/decoder. Let's go into detail about the Bitcoin transaction structure and look at its Binary Map [2]:



Picture 1 – Binary map of transaction

2. Proposed solution

We will use only standard inputs and outputs (P2PKH or P2WPKH) in our target transactions to make it possible to discard some constant values [3]. There are a lot of small constant fields. They are unnecessary to fit into SMS if we will rebuild the full transaction body after receiving SMS.



Picture 2 – High-level diagram

The sender and the intermediary must use the same technique of building the target transaction to make it possible for the intermediary to rebuild the full transaction body, which will be correct and valid for Bitcoin.

Table 1 – Target transaction structure

tx structure	**size (bytes)**	**how to transmit**
version	4	use constant
nLockTime	4	use constant
input_0		
prev_hash	32	block height + tx number (uint32 x 2)
index	4	as 1 byte (uint8 type)
signature	64	as 64 bytes
public key	33	recover from signature
nSequence	4	use constant
output_0		
address of Bob	25	as 20 bytes + 1 byte (output type)
amount to Bob	8	as 4 bytes (uint32)
output_1		
address of Alice (change)	25	use the same from the input
amount to Alice (change)	8	calculate based on the constant fee
total:	**192**	**97**

3. Explanation

For the version field and nLockTime field, we can use constant values on the sender side while creating a transaction. And we can use the same constants on the intermediary side while rebuilding the full transaction. So we do not need to transmit those values.

To transmit 32 bytes of prev_hash, we can use 8 bytes. 4 bytes to transmit the block height, where that transaction was confirmed, and 4 bytes – sequence number of that transaction in the block.

To refer to a particular output in the previous transaction, we can use a 1 byte unsigned integer, assuming that the transaction has no more than 256 outputs. So here, we can transmit via SMS only 1 byte instead of 4 bytes.

Let's consides scriptSig field as two separate fields: signature data itself and public key; discarding such constant values as SIGHASH flag. Assuming that we will always use the SIGHASH_ALL flag for all our transactions in that case [4].

The signature data can not be compressed or omitted, so we will transmit it via SMS in full size as 64 bytes. But the public key we can discard because an intermediary can recover it from the signature data and even check the correctness by hashing and comparing it with the address in the previous output.

The nSequence field can be omitted, assuming that the sender and intermediary will use the same constant value and will not exploit the functionality of alternative transactions like replace-by-fee.

Assuming that we will use only the most popular output types: P2PKH and P2WPKH; we can omit the full scriptPubKey by transmitting only the bitcoin address data and an output type as 1 byte value.

In our case, it would not be necessary to have the whole 8 bytes of the sending amount in satoshis because 4 bytes (uint32) is already 4294967296 satoshis at maximum value, which are 42.94967296 bitcoins. And we assume that the sender will not send more than 42 coins. So we can transmit 4 bytes instead of 8 bytes.

We propose not to transmit any data about change output but assume that it will be present in the target transaction. And the intermediary will rebuild the whole transaction, including change output, understanding that the sender will receive a change to the same address from the previous transaction and with a value calculated as the rest from sending amount and fee.

In our case, the sender and intermediary must use a constant amount of satoshis for the transaction fee to avoid transmitting it via SMS. In that case, we can also omit transmitting the change value because the intermediary can calculate it on his side:

$$change = spendingAmount - sendingAmount - constantFee$$

As you notice, we can "compress" our target transaction to 97 bytes, and that is fine because the maximum size of SMS is 140 bytes. The rest space of the SMS could be used to add a link to some website, where an intermediary can find tools and manuals on handling the data from that SMS. Or you can modify our proposal, e. g. transmit a new address for a change instead of using the old one.

4. Assumptions

The UTXO we are going to use for input of the target transaction must be fully confirmed in Bitcoin. Since the sender would not track the confirmation status of the target transaction and replace it if it is not confirmed. For example, in case of chain reorganization or some other transaction spent the same UTXO.

The change from the target transaction will go to the same address as an original UTXO. It is a slightly less secure and private approach, but we assume it is enough for our situation.

Transaction fee for target transaction will always be constant (e. g. 10'000 satoshis). It would be more expensive, but still the simplest solution and helps economize space in the SMS.

Supporting only the most popular input/output types: P2PKH and P2WPKH. It helps to avoid transmitting the custom Script via SMS. While we are going to support more than one type of output, we need to transmit one additional byte to determine the exact type (e. g. P2PKH or P2WPKH).

The intermediary can not steal the money, but the sender has to trust that the intermediary will be online, will receive SMS, and handle it. Otherwise, the payment will not occur.

5. Afterwords

That article is not an exact specification but rather one of the possible solutions for particular cases when the sender has an agreement with an intermediary to receive SMS, rebuild the transaction, and broadcast it to Bitcoin.

References

[1] https://ozekisms.com/p_2612-sms-character-set-handling-and-multipart-messages.html

[2] https://en.bitcoin.it/w/images/en/e/e1/TxBinaryMap.png

[3] https://en.bitcoin.it/wiki/Transaction

[4] https://en.bitcoin.it/wiki/Contract