Two Party double deposit trustless escrow in cryptographic networks and Bitcoin.

David Zimbeck
http://www.BitHalo.org

***Abstract*—Crypto-currency is a form of decentralized digital currency that has changed the world of finance over the past several years. Bitcoin[6] lacks a central authority and protects anonymity, while allowing a relatively low-cost alternative to fiat. It opens the doors for international exchange of commodities and has the potential to change how business is conducted. The signature and scripting system that Bitcoin uses allows for the creation of smart contracts. Also using signatures, it is possible to create accounts that require multiple signatures (multisig accounts) as well as transactions with multiple inputs and outputs. There has been discussion of some of the current weaknesses with smart contracts. We address these weaknesses to make smart contracts immediately accessible on the Bitcoin network. As proposed, this protocol offers a system of commitment schemes and business protocols that greatly reduces the issues of extortion and malleability from two-party escrow.**

Keywords: Bitcoin, cryptography, blockchain, smart contracts, escrow

I. Introduction

According to the Bitcoin wiki page for smart contracts [3], there are some drawbacks to the current protocol. Weaknesses with transaction replacement can jeopardize commitment schemes. Some of the proposals discussed such as "Secure multiparty computations on Bitcoin" [4] or atomic trading [9] suffer from a set of weaknesses which has prevented the early arrival of Bitcoin 2.0. First, transaction replacement has been disabled in the Bitcoin protocol. Second, transaction malleability can completely interfere with "future" transactions by mutating the transaction id (txid).

Bitcoins scripting system relies on a series of inputs and outputs. Each input is in a certain position, has a value and an address, and has a unique identifying txid number. The outputs can be one or many, and they are a list of addresses and values respectively. Signing is done using public and private key encryption thanks to Satoshi Nakamoto's brilliant solution to the "Two Generals problem" [6].

The scripting system has a variety of script hashes which were supposed to allow for a myriad of different contracts [8]. Our protocol does not actually rely on those although it is worth mentioning. Each input needs to be signed by any associated party and, in the case of multisig accounts, each party will sign their portion of each input.

The problem of not being able to replace transactions gives rise to many different extortion and double spend attacks. Additionally smart contracts suffers from a problem called "transaction malleability." Before a transaction is broadcasted to the cryptographic network, one of the parties will be in possession of the raw transaction. Although the signature itself can not be forged, it can, however, be mutated. By changing small bits of information, the transaction can be changed sufficiently to result in an entirely different txid. Although this does not jeopardize the security of the network it does make future commitments base on that id useless. Also if raw transaction data is exchanged before broadcasting, then one of the parties can attempt to submit early, invalidating some transactions.

Multisig accounts can be traditionally used as an escrow requiring 2 of 2, 2 or 3 or 3 of 3 signatures to confirm the spending of any input. A two party escrow (2 of 2 multisig) can suffer from the problem of extortion as well. A normal escrow (2 or 3 multisig) can result in collusion, bad judgment, and the usual problems which arise when entrusting third parties with your assets.

This protocol system solves these problems in a very simple and discrete manner which will greatly reduce the risk of loss, allow for trustless smart contracts, and allow trade between perfect strangers even if the parties themselves can not be trusted. Its implications and significance to the world are even beyond the scope of smart contracts and cryptographic networks. This paper forms an ideology and foundation for commitment schemes based on risk/reward combined with naive protocols. For the first time ever, untrustworthy parties will actually be compelled to perform in good faith.

II. The Protocol

A. The Agreement

For this example we shall assume two parties namely Bob and Alice wish to enter into an escrow to perform any type of contract. Perhaps Bob wishes to purchase bitcoins using cash from Alice, They decide to form a trustless two-party escrow.

First Alice generates a public (p1) and private key (pv1) pair for the prospective escrow. Alice sends a request to sell the bitcoins to Bob over the cryptographic network by sending, for example, the smallest possible payment she can of .00005500 bitcoins using a cipher that is open to the public. This cipher decodes a fake address attached as an additional output which was actually an encrypted personal message that reveals her BitMessage[2] address, and as a result, a secure communication channel can be established while at the same time circumventing spam. Now the parties are free to exchange the contract details without interference. Please note, Bob and Alice are not limited to communicating this way: They can employ any communication protocol they find suits them best. They may choose to use a custom public/private key protocol, BitMessage or e-mail. In the spirit of things, we choose to keep communication encrypted under BitMessage for security

purposes and to avoid ”eavesdropping.” Bob creates the hand shake with Alice by sending his new public key (p2) from a key pair (p2 and pv2), and the same transaction details along with his first txid which we will call tx1. He does not broadcast tx1 nor does he reveal the raw transaction details. Instead he sends the encoded hex which results in the first txid. Tx1 spends inputs of his choosing into a normal or multisig account to which he holds the keys. This is a ”temporary funding account.” The reason for the funding account is that Bob and Alice want to perform the transactions in a streamlined simultaneous fashion. They want to create a series of future transactions.

Ironically, in order to create a future transaction, you must have a signed transaction. Therefore, it would be impossible to generate the future txids without the uncertainty that the second party to receive a signed tx1 would not sign and submit it early. Thus, a temporary account circumvents the ”early broadcast problem”. Since the temporary account is not funded yet and Alice does not have the raw version of tx1, it would be impossible to submit anything to the network without Bobs approval.

B. The Creation Process

Using the public key that Bob sent, Alice generates the new multisig account (escrow). She creates tx2, which spends funds into her temporary funding account. She does not broadcast this either. She then writes tx3 that spends the inputs of tx1 and tx2 into the escrow. She signs this. She then creates a transaction that spends tx3 as 99.9% fees to the Bitcoin miners [5] to use as a penalty in case she breaks the commitment (timeouttx). This penalty is one which both Bob and Alice will suffer if they try to play fowl or cheat the deal. If there are reservations about collusion with miners or in the case of ”proof of stake” networks, then the funds can also be sent to a verifiable unspendable address such as 1BitcoinEaterAddressDontSendf59kuE. Alice signs timeouttx.

C. Malleability Refund

For this example Alice and Bob agreed that the risk of the party broadcasting the transactions was a high risk and could perhaps mutate tx3 to extort from the counter-party. Thus, tx4 is created by Alice. In tx2 Alice decided to send a larger amount to be destined for the escrow. She does not plan on keeping this as a deposit. Instead she writes tx4 which spends one of the inputs from tx3. In this scenario, tx3 pays to the escrow in two separate denominations scripted as separate outputs to the same addressone for her refund and the remainder for the escrow. Tx4 spends said input from tx3 back to herself as a reward for good behavior. Now, if Alice tries to mutate the txid of tx3, she also invalidates her refund. This creates a risk/reward scenario which can make the price of extortion unbearable.

One thing to also note here: A very common escrow will be where Bob and Alice have an equal deposit with the addition of Alice’s Bitcoin that was being sold as per their original agreement. So, they may be in escrow at an initial ratio of 2:1. This is by no means standard. They may also choose to perform micro-trading (sending many small payments instead of one large one) or any other form of contracting. However, if Alice is going to be in escrow for more than Bob, it would make sense for her to broadcast anyway. Therefore in that situation Alice and Bob may agree to forego tx4 altogether.

D. Finalizing the Escrow Deposits

Now that Alice has composed all the transactions, she signs them all and sends them to Bob. First she takes p2 and uses it as a cipher for an encrypted message. In this message she will send all of the raw transaction data with the exception of tx2. She can know that the message is secure since it uses p2 to conceal the data. This communication is not limited to the Bitcoin network, although that may be a good way to mitigate spam and automate the process. Rather, communication can equally be sent in any medium since the only person capable of decrypting it is Bob. In the case of BitMessage this may be redundant since BitMessage is theoretically secure. Bob receives this message and confirms it is from Alice. He then uses pv2 to decrypt the message. He now can freely review and sign everything. If Bob is happy with the agreement, he can use p1 to encrypt a message for Alice. Now that everything is signed, there are two copies of timeouttx. Thus, both parties can be secure during escrow negotiations since the risk is spread and structured according to their level of trust for each other. So now Bob sends every tx back to Alice including tx1. Now the transactions are all potentially valid.

E. Broadcasting and Managing Intercepted Transactions

Alice now uses pv1 to decrypt the final message and she then broadcasts all of the transactions with the exception of timeouttx. There is the remote possibility that a miner will be able to intercept the transactions and randomly mutate one of the txids.

The protocol includes two applications to mitigate problems which could arise from this. Firstly, multisig accounts also deemed ”p2sh” accounts have been described as more resistant and expensive to mutate [7]. Although I do believe this will require a lot of scrutiny, nonetheless my protocol uses p2sh accounts for every step of the transaction. Secondly, Bob and Alice will most likely be using a client that will automatically attempt to re-sign any of the broken transactions. If tx3 gets broken, Bob and Alice could take their coins back and attempt to sign everything again. If tx4 or timeouttx is broken, they will most likely re-sign them out of good faith. Furthermore and more importantly, the clients themselves are naive and will plan on deleting the escrow keys regardless of whether a destruction exists or not. Neither party can be sure that the counter-parties client will not automatically honor the protocol if violated. Naive multi-party protocols will cause a whole new wave in thinking about decentralized computing in the near future for this reason. It will be the basis for decentralized companies and software.

F. Finalizing Escrow

At this juncture, the funds are now in escrow. One of the variables in the contract was the time limit of timeouttx. This time limit is agreed upon in advance. It can be measured in hours, days, or quite possibly the blockchain itself. (The blockchain is the shared/public ledger of all Bitcoin transactions.) Bob and Alice can write any transaction from here.

They most likely will try to cancel or confirm upon the receipt of the commodity or completion of the agreement.

G. Taking Extra Precautions against Extortion

Both parties can take their own security one step further. They can choose to use ambiguous funding information such as a Western Union, bank deposit, gift cards, cash or prepaid debit cards. In the exchange of coins or commodities, there are several ways to conduct business without revealing the identity of the parties. One simple way is to use a public bulletin board or a BitMessage channel to post their contract to the world. This is especially useful for cash deals. Even in the case of certain outsourced employment contracts, it is reasonable to communicate in a filtered manner using an encrypted channel. As mentioned briefly in section 3.5, naive clients can communicate in a filtered manner with expectations of only certain types of information and commands thus completely obliterating theft and extortion. This revolutionary style of decentralized/filtered communication can be the basis for unlimited types of contracts and even decentralized Python programming and decentralized autonomous companies. Another way to reduce risk greatly to the point of being almost non-existent is by making a contract with small deposits at a ratio that is roughly 1:1. Then the contract is completed as a series of micro-payments or milestones. In this way, very large deals can take place with relatively small initial deposits. A special type of deal I refer to as ”micro-funding” would perform potentially unlimited maneuvers under this contract and can easily fund million dollar deals in a trustworthy risk-free manner. A ”contract bridge” can be created in the same way for virtually any application. Try to think of it as a form of self-insurance.

Regardless, the proposal here is to mitigate risk further by either filtering or eliminating extraneous information partially or entirely. If Bob and Alice have nothing else to discuss but the raw contract data itself, it makes threats impossible.

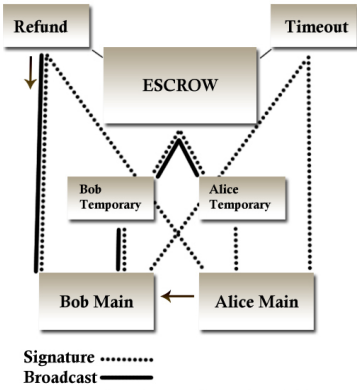

Fig. 1. Illustrating of the proposed protocol. Refund and timeout are signed using Bob’s and Alice’s escrow private key. Everything is signed in advance. Alice sends all her raw transactions to Bob to broadcast.

III. Implementation

This protocol has now been tested using a home made program originally coded in python called BitHalo [1]. The program itself is completely new, and is being published the same time as this paper. Being open source, it should be easy to review, audit and expand on the various different permutations and methods of implementation of the protocol. This is only the foundation and new more advanced protocols will undoubtedly be proposed as a result of this paper. There is technically no limit on the ways ”Smart Contracts” with two deposits can be applied. The wallet addresses for BitHalo can be reviewed by looking at the associated wallets addresses over the Bitcoin blockchain explorer blockchain.info [10]. Open source software is gaining ground, and we believe that these concepts will evolve and expand at an ever increasing rate.

IV. Conclusions

The protocol has been described in the simplest terms. We believe the result will be creation of a new paradigm in finance where people will no longer have to depend on untrustworthy third parties.

There was an example of a Bitcoin exchange that ”lost” the equivalent of hundreds of millions of dollars. This behavior is unacceptable, and it costs the consumers dearly. Furthermore escrows in real estate transactions have been known to go south and complete disappear. Banks have been known to disappear throughout history as well. In fact, almost every sector of economy that involves third parties runs the risk of loss to the consumer. This protocol brings a whole new set of solutions back into the hands of the individual where they now have full control of who they decide to trust and how they decide to structure that trust. Additionally this brings confidence into the hands of young businesses that want to find a way to earn the trust of their clientele. It is a very attractive proposition in any employment contract knowing that both parties employee/employer made an advanced commitment.

This also allows the creation of a bartering system that is insured by the deposits themselves. Under the bridge of trustless contracts, two parties can perform potentially unlimited maneuvers under the same contract. Two complete strangers who were originally untrustworthy are now capable of building trust despite their background, however dubious it may or may not be. Our society relies on trust to conduct business, form relationships, and exchange commodities. Finally after all these years and millennia of waiting a proposal has finally come to fruition. We hope this helps evolve an ever evolving economy into one that is more secure from uncertainty.

References

[1] BitHalo. http://www.bithalo.org. 2014.

[2] BitMessage. https://bitmessage.org/wiki/main_page.

[3] Bitcoin Contracts. https://en.bitcoin.it/wiki/contracts.

[4] Daniel Malinowski Lukasz Mazurek Marcin Andrychowicz, Stefan Dziembowski. Secure multiparty computations on bitcoin. *IACR Cryptology ePrint Archive*, page 784, 2013.

[5] Bitcoin miners. https://en.bitcoin.it/wiki/mining.

[6] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[7] Salvaging refund protocols from malleability attacks using p2sh. http://www.likelyanswer.com/10425812/salvaging-refund-protocols-from-malleability- attacks-with-p2sh.

[8] Script. https://en.bitcoin.it/wiki/script.

[9] Atomic Trading. https://en.bitcoin.it/wiki/atomic_cross-chain_trading.

[10] David Zimbeck. Bithalo tests on blockchain.info: https://blockchain.info/address/349hfhbui8x7d7uvaxklxfjch9fgovx7at.