Blockchain Certification Protocol (BCP)

Fu Yong Quah
fuyong@fyquah.me
www.fyquah.me

Abstract. A semi-decentralized certification system built above the existing [1]Bitcoin network, an innovative peer-to-peer payment network. Current certification system requires an eyewitness to act as a trustworthy party. Frauds, such as cases where the eyewitness is bribed a party of a certificate, may result in complications. The proposed certification system still requires a third party to act as an eyewitness of the signature of a contract. However, involvement of any third party will not be required in verification. Contracts or certificates that can be solved can involve an arbitrary number of parties. Verification of such contracts can be made on an individual basis, which means presence of individual signatures in a certificate or contract can be identified independently.

0. Terminology

Bitcoin	An innovative peer-to-peer payment network without a central authority.
Blockchain	A transaction database shared by all nodes in the Bitcoin blockchain based on the Bitcoin Protocol.
Marker Output	The OP_RETURN output in a Bitcoin Transaction. The marker output will be used to store information in this protocol.
Certification Node	A node in the bitcoin network which has announced that he will certify certificates or contracts using this protocol.
Rights Output	A Bitcoin transaction output which represents the right to broadcast transactions on behalf of a certification node.
Parent Output	A special kind of rights output, which gives rights to destroy existing child outputs or spawn new child outputs. At a single instance, there can only be one parent output for a certification node.
Child Output	Non-parent rights output. Holders of these outputs are not allowed to spawn any further childs for the node.
BCP-Transaction	Refers to a transaction that is aware of the Blockchain Certification Protocol. This transaction abides to the strict protocol rules in this document.

1. Introduction

This semi-decentralized system is built on top of the Bitcoin Network for its reliability of information storage. [2]With the introduction of the OP_RETURN OP_CODE in the release of Bitcoin 0.9.0, a good future of widening the usage of bitcoin beyond a payment system has been discussed. [3]The usage of Bitcoin to represent assets other than the currency itself has often been reckoned as the possible future of such currencies. Blockchain Certification Protocol is a system that exploits the usage of this OP_RETURN on top of the current Bitcoin Network.

Such concept of certifying the existence of a document at a certain point of time has been explored by concepts such as [4]Proof of Existence (PoE), where the SHA256 hash of a document is broadcasted as a transaction to the Bitcoin network. While PoE manages to address the problem of the existence of a document, it fails to identify whether a certain individual has actually view and agree with the context of the document. The fundamental problem of contract signature cannot be solved using only PoE.

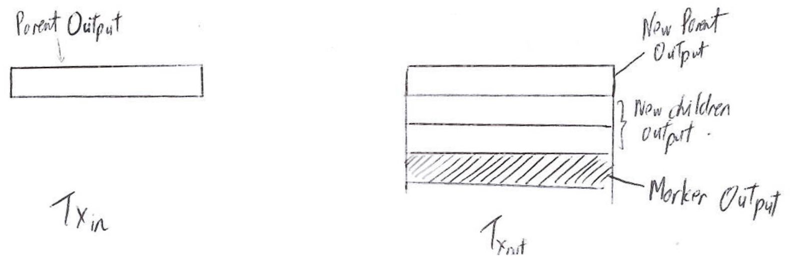
We propose to solve this problem by inserting the digital signature of documents in the blockchain, instead of only the hash. By using a secure and compact cryptographic signature algorithm, namely the ECDSA, we can partition and store a single 64-byte ECDSA Digital signature in 2 Bitcoin transactions. The binding of a certain Bitcoin address to a certain ECDSA public key will require some protocol and proper key management. This document will explore a systematic approach to address this issue in an elegant manner.

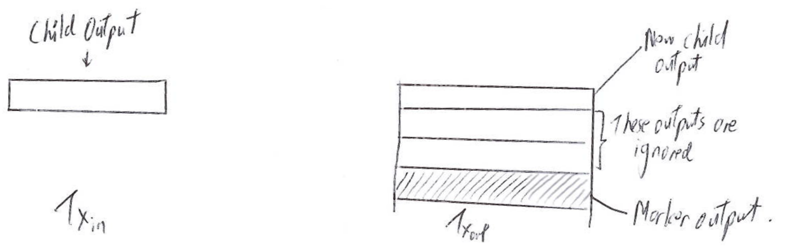
2. Certification Nodes

Initializing a Certification Node
A node will broadcast a Bitcoin transaction to inform to other nodes that he is starting to certify certificates. Details of the structure of the Bitcoin transaction will be further explored in a later section. The outputs up to but not including the market output represents rights outputs, while the first output is the parent output. The rights output reference to the node that has just been created. [5]The concept of labeling outputs of Bitcoin transactions is inspired by colored coins.

Retaining of Rights Output after Transactions
An ordinary Bitcoin transaction is considered as a BCP-transaction if and only if the first input of the transaction is a previously generated rights output and the marker output obey certain specifications of this protocol. As transaction inputs are spent in a transaction, its outputs are labeled as new rights output appropriately based on the type of output that was supplied in the input.

If the supplied input was a parent output, all the outputs up to but not including the marker output will be valid rights output for the node. However, only the first output will be a new parent output; the others are simply new children outputs. This is consistent with the idea that only one parent output per certification node can exist at any point of time. On the hand, if a child output was used for the transaction, the first output before the marker output will be a newly generated rights output while the remaining output between the first outputs and the marker output will be ignored. This means that the holders of child outputs will not be able to spawn further children. In both cases, if the transaction does not provide any outputs before the marker output, it signifies the destruction of a rights output. We will call such transaction a terminating transaction. This can be useful to destroy a child or termination of a certification node. The idea is illustrated in Figure 1.1.





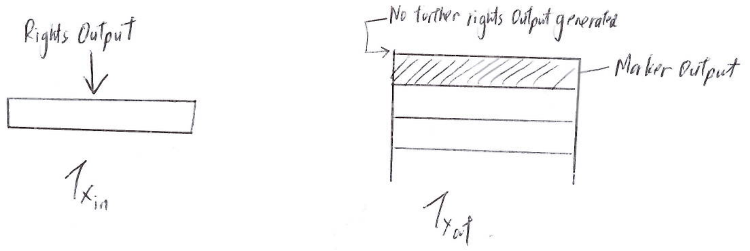


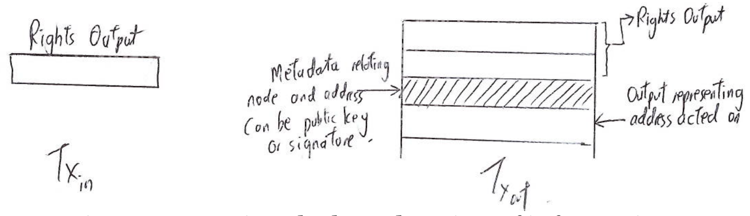
Figure 2.1: from top to bottom: Parent Output BCP-transaction, Child Output BCP-transaction and terminating BCP-transaction.

The ability to spawn new children for a certain certification node is applicable in a practical sense. If a website which runs the certification service, he might find it useful to have a few parallel Bitcoin nodes using children outputs to serve his clients simultaneously rather than having a single node dealing with all his clients in a synchronous manner.

To identify that node associated to a rights output, one can recursively trace back the transaction that generates a particular output all the way to the genesis / initialization transaction.

3. Signature Mechanism

Directing a Transaction to a Particular Address
Before introducing the signature mechanism, it will be important to address how a transaction is directed to an address. This is because a single Bitcoin transactio is not directed to a single address (as opposed to conventional payment systems). Hence, we will define the address directed to by the BCP-transaction is the first recipient address of the output directly after the marker output. Other addresses and outputs will be irrelevant in our use case. Those inputs and outputs act as source of funds and change outputs respectively. An illustration of this concept is provided in figure 3.1.



Rights Output
Tx_{in}
Metadata relating node and address Can be public key or signature.
Rights Output
Output representing address acted on
Tx_{out}

Figure 3.1: A transaction representing the broadcasting of information to a Bitcoin address by a certification node.

Binding a Bitcoin Address to a Public Key
Binding a user to a public key can be referred more commonly as "registering" a user or individual to a certification agency. In our protocol, it is equivalent to binding a Bitcoin address to a 33-byte compressed ECDSA Public key in the scope of a certification node. In other words, all signatures made by this Bitcoin address and broadcasted by this node will have to be verifiable by this public key. The mechanics of broadcasting this transaction is straightforward: The certification node broadcasts a BCP-transaction with the marker output containing the 33-byte public key.

A possible fraud from this scheme is false identification. For example, consider the case where Alice publicly publishes her Bitcoin address to her blog (to accept donations) and Eve notices such address. She then initializes a node and falsely binds that address (Alice's address) to a public key (hence, she knows the private key). Eve can start making false signatures at various contracts and certificates, while the world might mistake Alice as the signer of those documents. The solution to such fraud is the enforcement of a strict rule where the user has to explicitly announce the binding of an address to him/herself. At the same time, committing of fraud by certification nodes in cases as such will reduce the confidence of the general public on those nodes. As long as there are sufficient honest nodes in circulating in the network, such problems will be negligible.

Broadcasting a Signature
A user of a certification node shall sign the document and obtain a signature. This user will then submit the hash (any reasonably collision-resistant hashing algorithm, for example: SHA256 should suffice) of the document to the certification node. The node has to firstly verify the signature of the hash with the user's public key (it is publicly available in the Bitcoin Blockchain). Once it is verified, a BCP-transaction, which contains the signature (32-bytes in length) in the market output, is broadcasted to the user's address. The signatures generated by any particular address are *not anonymous,* although the contents of the original document cannot be read off the Blockchain. This means that although we do not know what documents a certain individual has signed, we do know the number of documents he has signed. Such trade-off is made so that verification, where most conflicts and disputes arise, can be made smoothly without the need of a trustworthy third party.

Verification of Signature
Bob wants to prove that Alice has signed a certain contract in the past. All he has to do is to identify the certification node that was used by Alice and Alice's Bitcoin address registered to that node. Once he has identified these 2 components, he can easily verify the signature. If he has a copy of the signature, he can simply verify that signature with Alice's public key and check if the signature exists in the blockchain. Otherwise, he can carry out an exhaustive search on Alice's entire signature space to check which can be verified along with the document by Alice's public key. Such searches are reasonably feasible as the search space is much smaller than the size of the set of all possible signatures.

4. Transaction Format

Fundamentally, all BCP-transactions are in fact bit coin transactions with stricter specifications. To reduce the chances of running into the risk of collisions unrelated transactions, the marker output of BCP-transactions shall observe a strict format. Using the 40-byte capacity of the marker output, it will be allocated as shown in figure 5.1.

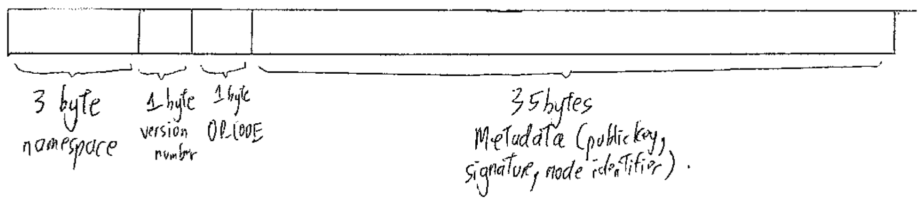


Figure 4.1: Allocation of Marker Output bytes in BCP-Transactions

The proposed 3-byte namespace is "BCP" (hex: 0x42 0x42 0x50), the abbreviation of Blockchain Certification Protocol. The usage of a version number provides room for improvement in the future. The main required op_codes include, but are not limited to:

- Initialization of node
- Binding user to public key
- Publishing the R signature or a user
- Publishing the S signature of a user
- Do Nothing (useful for spawning new children)
- Termination of the whole node or children

It maybe noticed that unlike the Bitcoin Scripting Language, where the length of arbitrary metadata is specified, the length of the data is not specified in our marker output. This is because such length can be easily deduced from the length from the marker output, which is specified in the marker output of the Bitcoin transaction itself.

As briefly mentioned in section 2 (Certification Nodes), all rights output will have to be the first input of the BCP-transaction. Such restriction will ease the software in determining the node affected by a certain transaction. Similiarly, the first output of the transactions outputs will refer to an output of the same tier as the input provided, whereas the subsequent outputs up to but not including the marker output refer to an ouput of a lower tier than the input provided. This example is clearly illustrated by using parent and child outputs in section 2.

In the case a node is broadcasting a transaction to a user (when the node wants to publish a public key or broadcast a signature), the rules above apply, in addition to having the target address as the first address of the output directly after the marker output.

In the case of our certification protocol, the amount of transaction will not incur any significant changes to our verification model. Hence, the amount of every transaction is not important. However, imposing a transaction fee and a certain minimum amount to every output will be necessary to decrease the time taken for the transaction to be mined.

5. Applications

The semi-decentralized of this protocol opens up a lot of opportunities in many existing spaces where contracts and certificates are often used. Some real world examples include:

- *Software* – A software company can release a software patch and publish the signature of the hash of the patch using this protocol. Users can then verify the integrity of the software patch using this protocol as well.
- *Education* – A college or educational institute can sign every certificate they issue so everyone can easily verify the integrity of a college certificate.
- *Housing* – A tenant and a landlord may sign a housing contract with certain negotiated term and conditions. That signature can prove the agreement of certain terms in the event of a dispute..
- *Share Certificates* – A company can issue share certificates to investors. Both parties can use these certificates to prove ownership whilst digitally managing their certificates.

With careful analysis, we can categorize most applications into two main types – certificates and contracts. We shall define certificate as a document that is issued or published by an individual to the rest of the world, and should be easily verifiable by anyone. We can define contracts as documents that represent an agreement between two parties and shall be verifiable by anyone who has an unaltered copy of the original document. Both concepts are illustrated in figure 5.1 and 5.2 respectively.

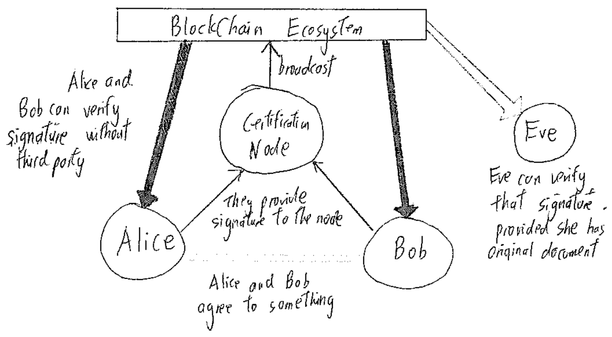

BlockChain Ecosystem
broadcast
Alice and Bob can verify signature without third party
Certification Node
Eve
They provide signature to the node
Alice
Bob
Eve can verify that signature, provided she has original document
Alice and Bob agree to something

Figure 5.1: Usage of the protocol for contracts

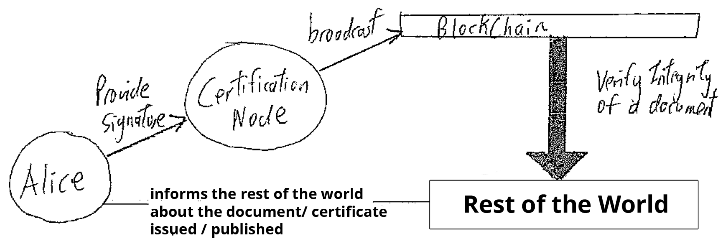

broadcast
BlockChain
Verify Integrity of a document
Provide Signature
Certification Node
Alice
informs the rest of the world about the document/ certificate issued / published
Rest of the World

Figure 5.2: Usage of the protocol for certificate

From an entirely technical perspective, both usages involve similar signature mechanism. The main difference lies in the handling of information. In the case of contracts, such information is obviously private and should be kept safely by both parties of the agreement. In the case of certificates, the information being signed is public and can be safely released to public. The use case of the protocol for certificates is can be technically described as the usage of cryptographic hash function for message integrity along side with the application of Blockchain in it for long term reliability.

6. Conclusion

Throughout this document, we have explored a semi-decentralized solution and protocol to provide a contract and certificate verification solution, which requires no third party in verification, leveraging on the reliability of the Bitcoin network. This protocol can be implemented directly on the Bitcoin Blockchain. By using a rights output based model to represent nodes in broadcasting signatures or public keys, it is possible to divide the tasks of a single certification node to several children, and hence, allow parallel processes. With the presence of a single parent output and various children outputs, it is possible to simultaneously run many certification processes while having a single authority of within a node to manage the operations of the output.

Appendix 1: OP_CODES Implementation Reference

OP_CODE	**Description**
0x00	Initialization of Node
0x01	Binding address to a 33-bit compressed ECDSA Public Key
0x02	r-coordinate of a signature
0x03	s-coordinate of a signature
0xE0	Do nothing (useful for spawning children)
0xF0	Destroying a child node
0xFF	Termination of the node as a whole

Appendix 2: Relevance to SideChains

[6]The ability to incorporate sidechains to issue new currencies on top of the current Bitcoin network suggests a relevant improvement to the Blockchain Certification Protocol. In the case of BCP, although we are not using currencies, a good reason to explain the prospective of sidechains include, but are not limited to:

Having a proper currency to represent rights to a certification node

The current state of the protocol relies on the recursive search of rights output to identify the node it has rights to. Using a unit of currency that contains clear labels on the node it belongs to can potentially save computational time in identifying the node itself.

Remove the need of having unnecessary inputs and outputs

Building this protocol above the Bitcoin blockchain require us to abide to two rules – The rules of Bitcoin transactions and those of our protocol. As a result, the transaction may contain a lot of redundant inputs and outputs, acting as additional funds and change respectively.

A more systematic way to insert metadata relying on marker outputs
As OP_RETURN only permits the insertion of 40 bytes, signatures require 2 transactions to broadcast. By creating an entirely new transaction format in the sidechain, we can definitely explore the possibility of including metadata as a native component of our sidechain's transaction.

Unlike most traded assets, these rights output not only hold not real-world value, but also cannot be given as a miner's reward or transaction fees (as it represents the rights to a certain node, something definitely not publicly distributable). Without any incentives, members within the network will be reluctant to contribute computational power to support the network.

We proposed a solution that introduces two forms of currency into the network. The first form of currency represent rights to broadcast metadata on behalf of a certification ode whereas the second form of currency represent tokens. In other words, to broadcast a transaction, a certification node would require a certain amount of tokens, which act as mining fees. These tokens hold value and can be converted into Bitcoins of equivalent value, giving miners in our sidechain the incentive to contribute processing power.

Reference

1. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. *Consulted*. 2008;1(2012):28.

2. Bitcoin client version 0.9.0 README. https://bitcoin.org/bin/0.9.0/README.txt. Updated 2014.

3. the Economist. Bitcoin's future: Hidden flip side. http://www.economist.com/news/finance-and-economics/21599054-how-crypto-currency-could-become-internet-money-hidden-flipside. Updated 15 March 2014.

4. Araoz M. **What is proof of existence?**. http://www.proofofexistence.com/about. Updated 2014.

5. Rosenfeld M. *Overview of colored coins*. 2012.

6. Back A, Corallo M, Dashjr L, et al. *Enabling blockchain innovations with pegged sidechains*. 2014.