

BBc-1 : Beyond Blockchain One

- An Architecture for Promise-Fixation Device in the Air -

Kenji Saito and Takeshi Kubo
{ks91|t-kubo}@beyond-blockchain.org

Revision 0.2 – December 11, 2018

1 Introduction

1.1 Background and Overview

Blockchain technology today has some sustainability risks including cryptographic techniques being used becoming obsolete and the price of native cryptocurrency declining (thus miners and/or validators get incentivized to leave). There are also other technological problems and problems of technical governance, involving political situations surrounding development communities and miners/validators.

BBc-1 (Beyond Blockchain One) is a kind of “middleware” to give long term solutions to these problems, and at the same time to provide support for currently ongoing application development. It is designed to implement requirements specified in “BBc Trust”¹, and is being developed as a set of open protocols and open source reference software that implement them, obtainable and usable for free². We are in the process of forming a global open community, *beyond-blockchain.org*, a non-profit organization of corporations and individuals who will keep working on this project together.

1.2 Basic Design Principles

Figure 1 shows an overview of the BBc-1 architecture.

Today, major applications of blockchains can be categorized into two: currencies (systems for transferring quantities) and assets (systems for provenance and managing rights). BBc-1 will support both, and provide API (the interface between the ledger core and the application layer) and SDK (set of libraries that run within the application layer) for these.

Notably, BBc-1 provides the following facilities for application development:

¹Our charter. Found at <https://beyond-blockchain.org/public/bbc-trust.pdf>

² GitHub repository is found at <https://github.com/beyond-blockchain/bbc1>

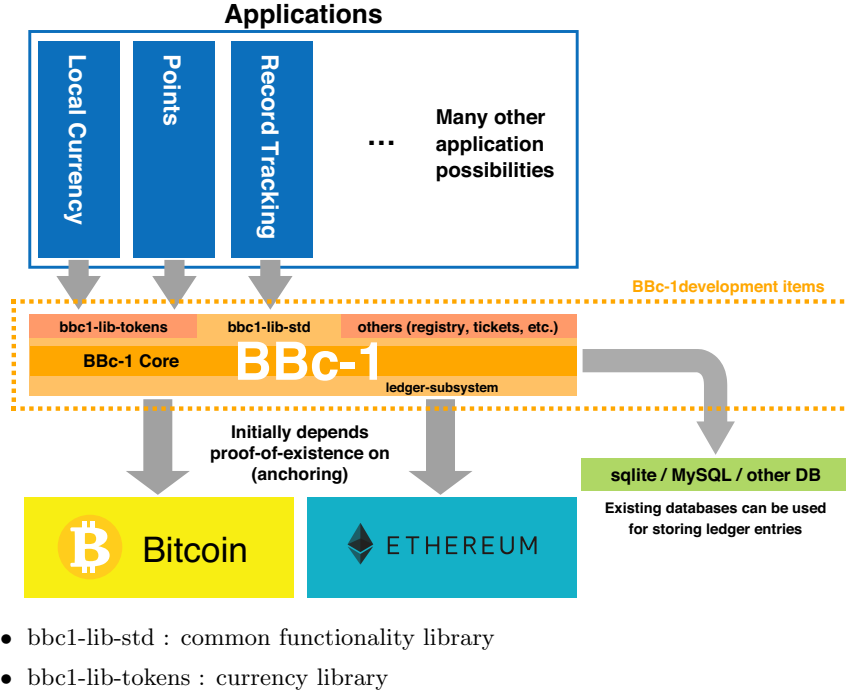


Figure 1: BBc-1 Architecture

1. Ease of design through larger freedom of expressing relations among information.
2. Commitment of transactions is made through consensus among concerned parties instead of third-party validators.
3. Improved detections of falsification.

BBc-1 provides proofs (or disproofs) of transactions to applications. Initially, it is done by utilizing proof-of-existence functionality of existing blockchains. A blockchain to be used for this purpose can be dynamically chosen.

The abstraction layer for blockchains will evolve so that it will work without underlying public ledgers in the near future. In the end, we aim to replace blockchains themselves with BBc-1.

2 Understanding Blockchains

2.1 Functional Layer Model

What blockchains or other kinds of distributed ledgers do can be understood in terms of functional layers as follows (in the lower-layer-first order):

1. Guarantee of Validity (of transactions)

- The ledger guarantees that a new transaction cannot be mutated, does not contradict with the existing history of transactions, and is committed by a user or users who have right to do so.
- The ledger also guarantees that nobody can stop a user or users who have right to do so to commit a new valid transaction.

2. Proof of Existence (of transactions)

- The ledger provides the proof of existence of a transaction.
- The ledger does not allow anyone to delete the evidence of a transaction committed in the past, or to fabricate an evidence of a transaction that has never been committed in the past.

3. Consensus on Uniqueness (of transactions)

- If two contradicting transactions were to be committed, all users of the ledger (will eventually) see the same one of them in their views of the correct history of transactions.

4. Descriptions of Rules (that define semantics of transactions)

- The ledger allows users to define semantics of transactions. (In Bitcoin, all transactions are basically about sending bitcoins.)

By providing these functions, a blockchain or any ledger system can act as a “promise-fixation device in the air”, in which nobody can deny the content or existence of committed promises or records whose validity anyone can verify, and nobody can stop legitimate users to commit or verify such promises or records.

The true value of such a device is that it provides proof that a digital signature situated in the past is either valid or invalid.

2.2 The Last Will Test

The Last Will Test can test whether a designed ledger system such as blockchains can completely function as a “promise-fixation device in the air”. Here is what the test asks:

Can your ledger system be used for saving a person’s last will and testament, so that you can provide proof for all interested heirs that the document is kept as-is after digitally signed by the person, without asking them to *trust* you?

This test is meaningful because the private key is no longer private after the person dies, and there generally is a reasonable doubt that a heir or heirs has found the key, rewritten and signed the document with the found key, and replaced the stored document with support from you, the operator of the ledger system.

Many private or consortium ledgers would fail this test (unless all heirs are members of the consortium), because they do not provide external proofs.

Many public ledgers, on the other hand, would pass this test for the time being, but they may stop functioning at any time due to some external reasons mainly concerning the prices of their native cryptocurrencies.

Our aim is to design BBc-1 as a ledger system to pass this test beyond reasonable doubt, and provide a reference implementation of the software.

3 Features

We will describe features of BBc-1 in terms of each layer above.

3.1 Guarantee of Validity

BBc-1 uses a data structure similar to so-called UTXO (Unspent Transaction Output) structure used in Bitcoin and alike, because of its ease of understanding and of accounting, but the data structure also allows state-tracking. Therefore, both UTXO and state machine approaches, two common basic structures of ledger systems, can be used with BBc-1.

To support confidentiality of transactions, BBc-1 works over inter-connected multiple domains of networks, where the content of transactions, including identities of involved parties, is visible only within each domain (digital assets handled in transactions can be encrypted to further support confidentiality).

Figure2 shows the structure of a BBc-1 transaction data, where

Header section contains meta-data of the transaction such as timestamp,

Events section can contain set of operations over assets as outputs of the transaction, and also specifies the parties who can perform further operations over the assets,

References section can contain references to past events and indications of <signature, public key> pairs in the signatures section as the proof of identities,

Relations section can contain general references to past related transactions,

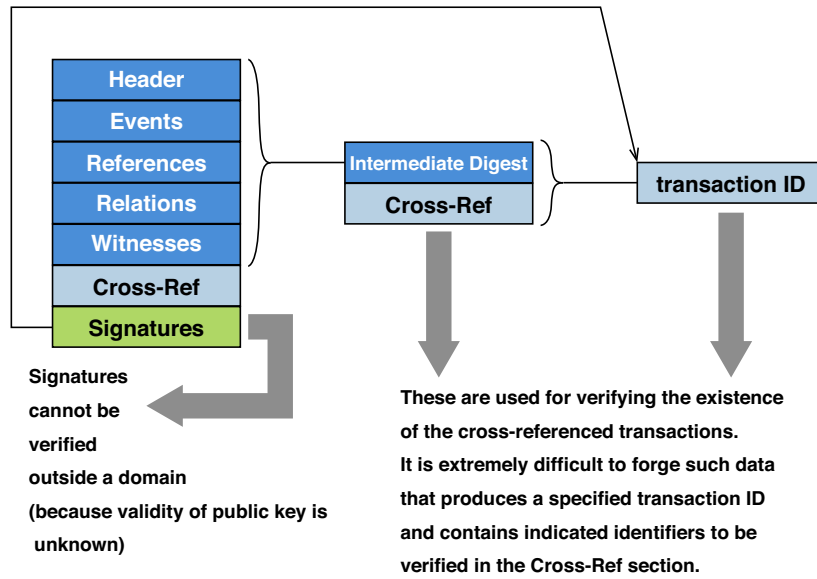


Figure 2: Structure of a BBc-1 Transaction Data

Witnesses section can contain identifiers of designated signers for this transaction,

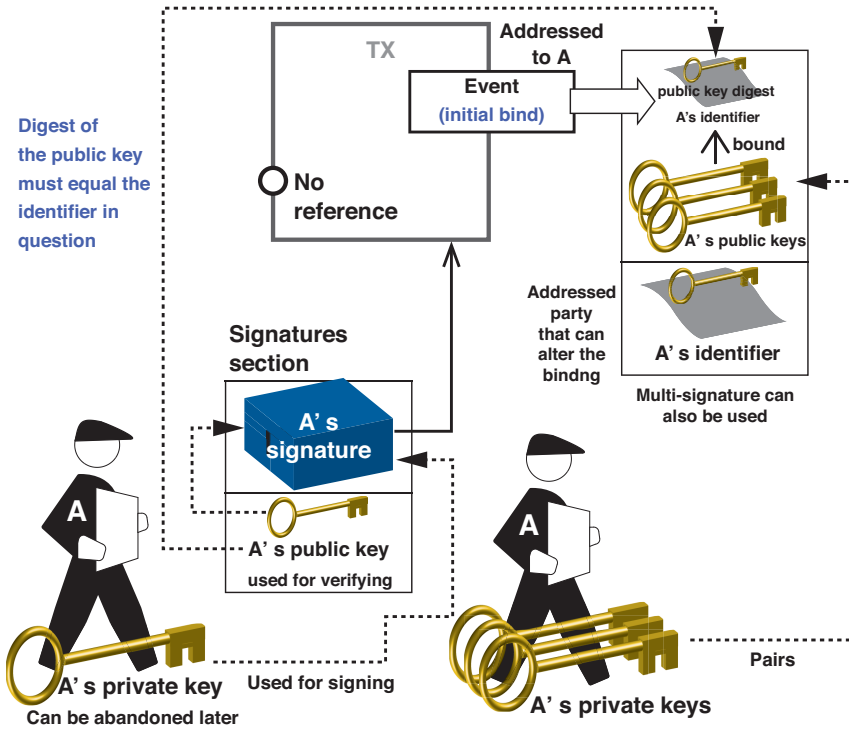
Cross-Ref section can contain the identifiers of transactions (or set of transactions) from different domains that this transaction provides proof of existence, and

Signatures section contains a set of <signature, public key> pairs that sign the identifier of this transaction.

Blockchains today generally use the digest of a public key as an identifier of an external actor. This means that if the corresponding private key is lost, proving the identity of the user by digital signature will no longer be possible, and the user loses control of coins or assets forever.

BBc-1 advances solutions to this problem by separating identifiers and sets of public keys, as shown in Figure 3, for example (it is applications' responsibility how this separation is actually designed).

In this example, which is implemented in *bbc1-lib-std*, or the common functionality library, the binding between an identifier and a set of public keys is stored in the ledger itself (therefore, the binding is visible only within the domain, and the signatures cannot be verified outside the domain). When verifying a digital signature, a verifier sees whether the specified public key is bound to the addressed identifier on the ledger or not.



* An identifier is the digest of a public key, whose pairing private key must be used for signing the TX that initially binds the identifier and public keys.

Figure 3: Separation of Identifier and Public Keys (in bbc1-lib-std)

3.2 Proof of Existence

At least in the initial stage of development of BBc-1, a proof of existence of a transaction is performed by checking the root of a Merkle tree written onto a public blockchain as illustrated in Figure 4, as commonly practiced in blockchain applications.

However, problems of proof of work or proof of stake, methods used in public blockchains for protecting such systems from undetected falsification, are that their safety and liveness depend on the market values of their native cryptocurrencies paid as rewards and/or used for reserving voting rights.

Instead of these techniques, we will use proof of context, or a cross-reference method in which a transaction provides proof of existence to a set of non-related past transactions in an unrelated domain by containing their identifiers in the cross-ref section of the transaction data (Figure 5). This type of technique is often referred to as *DAG*³, because it forms a

³We believe that this is a misleading term, because directed acyclic graphs are seen everywhere in the structures of a ledger system.

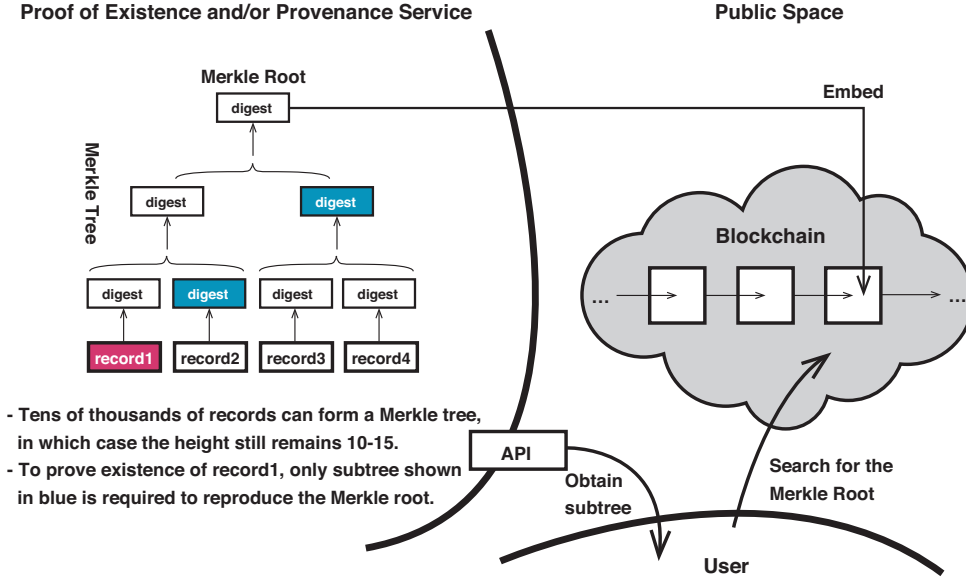


Figure 4: Proof of Existence Scheme using a Blockchain

directed acyclic graph of reference relations. In BBc-1, these cross-references are performed among transactions in unrelated domains (and thus among different contexts), to minimize possibilities of colluding to alter records.

Moreover, by having timestamp services trusted and chosen by each application periodically committing time-defining transactions, general transactions can be proven to have existed in some bounded real-time in the past. In return, those timestamp services can be protected against fraud themselves by taking part in the system of proof of context (again, Figure 5).

3.3 Consensus on Uniqueness

It has been well known by now that blockchains do not actually achieve consensus, through some enlightenment efforts or real cases and risks of hard-forking events and block-withholding attacks, for example.

This problem, we believe, is due to a design fault in which possibility of consensus among undefined participants is pursued, which we believe is impossible.

In Bitcoin, for example, coins are purely assets, without being backed by any debts. This means that when a coin is double-spent (thus the value is copied and doubled), it is unclear who is at a disadvantage. This in turn requires such a design that the whole participants as a group must agree on the uniqueness of transactions, which we know is impossible because we

3.4 Descriptions of Rules

In the future, BBc-1 may allow users to define *smart contracts*⁴ with languages designed for that purpose, but the mechanism for it is now in the process of being designed.

At least in the initial stage of development of BBc-1, applications will have to include their own logic to define and interpret the semantics of transactions they handle. We provide an SDK for the purpose.

3.5 Networking

Detailed descriptions of our intra-domain and inter-domain networking will appear.

4 Design Tasks

The following is the list of design tasks for us to work on:

- Enrichment of the SDK that include the common functionality including identifier-public keys separation, applicable replication techniques for redundant signing actors, etc., digital currencies, general ticketing, and general registry tasks.
- An efficient way to confirm the proof of existence using proof of context, or the cross-reference technique.
- Mechanism and languages for smart contracts (running program code as an asset).
- Incentives for nodes to form a domain to provide proof-of-context services (probably better not to depend on issuing coins; but rather a tit-for-tat mechanism that disallows BBc-1 proof-of-context services to those who do not really participate in the inter-domain network).

Revision History

Rev.0.1	2017-10-31	Initial revision
Rev.0.2	2018-12-11	General update to accommodate ongoing design

— End of Document —

⁴ A smart contract is a piece of program code whose validity and existence is assured so that users can be certain that it has not been modified after deployment.