Formal Expression of BBc-1 Mechanism and Its Security Analysis

Jun KURIHARA[†] and Takeshi KUBO[††]

[†]kurihara@ieee.org
[††]t-kubo@zettant.com

October 31, 2017

1 Introduction

Bitcoin and its core database/ledger technology Blockchain [1] have been capturing both industries and academia by its sophisticated architecture to guarantee trust and unforgeability of data records with no central authority. It has been enormously investigated as an alternative to the current centralized database system and a potential technology of new type of future information technology services.

As mentioned at the head of this section, Blockchain was proposed as a building block of a cryptocurrency Bitcoin. Hence, in the case where we use Blockchain as just a trusted database, participants of a Blockchain instance must follow unreasonable policies of Bitcoin with undesirable resource consumption, e.g., mining, even if it is completely unnecessary. Observing such facts, Saito et al. introduced a new distributed unforgeable database/ledger called Beyond Blockchain One (BBc-1) in 2017 [2] as a skeleton over which new kind of applications can run.

The design principle of BBc-1 is to guarantee only the following one statement: "any information must be registered in such a way that users can authenticate the information itself and its registrant." This means that verification of both registered information and registrants is simultaneously enabled through BBc-1.

Our aims of this paper are (1) to formally define the flow and mechanism to register the data to an instance of BBc-1; (2) to analyze the security of BBc-1 and show its both advantages and disadvantages (restrictions); and (3) to provide a design hint/guideline for applications considering the above (dis)advantages. Here we should emphasize that BBc-1 was designed so as to provide just a simple skeleton, and that the restrictions of BBc-1 itself should be treated in the application layer according to its desired security level and design principle.

2 Formal Expression of Transaction in BBc-1

2.1 Basic Flow to Generate Transaction Data

In this subsection, we give several definitions and formally describe the mechanism of BBc-1. Basically, BBc-1 can be viewed as a function to generate data objects called

transaction data and register them to the network nodes in a distributed manner. Here we simply call by a *transaction* a operation to generate a transaction data object, and note that one transaction generates only one transaction data.

First, without loss of generality, we suppose that every transaction in the world of BBc-1 can be serialized in chronological order, and an integer $i > 0$ is regarded as the universal indicator that indexes the i-th transaction. Let Φ_i represent the i-th unique *transaction* that takes a set of data blocks $\mathcal{A}_{in,i}$ and newly generates another set $\mathcal{A}_{out,i}$ as follows.

$$\mathcal{A}_{in,i} \overset{\Phi_i}{\longmapsto} \mathcal{A}_{out,i},$$

where $\mathcal{A}_{in,i}$ and $\mathcal{A}_{out,i}$ are called sets of *assets*. We should note that objects in $\mathcal{A}_{in,i}$ are ones generated by past transactions Φ_j's $(0 < j < i)$, and that objects in $\mathcal{A}_{out,i}$ will be taken by future transactions Φ_k's $(k > i)$. Also note that $\mathcal{A}_{in,i}$ could be an empty set if $\mathcal{A}_{out,i}$ is a set of newly produced assets and Φ_i is a transaction of their initial registration.

Let T_i be the transaction data object generated by the transaction Φ_i. T_i consists of two parts, the data part C_i and the signature part S_i, that is,

$$T_i \triangleq C_i || S_i.$$

In the following subsections, we consider how transaction data objects T_i's are generated and registered in the world of BBc-1.

2.1.1 Generation of Data Part C_i

Although the entity who registers a transaction data object could be any participant of a BBc-1 instance, it is typically a participant executing the transaction or a trusted third party. Recall that the purpose to register transaction data objects in BBc-1 is to verify the trust and existence of the transaction Φ_i transforming $\mathcal{A}_{in,i}$ into $\mathcal{A}_{out,i}$. Hence in BBc-1, we shall register the mapping $\mathcal{A}_{in,i} \overset{\Phi_i}{\longmapsto} \mathcal{A}_{out,i}$ and a set of participants $\mathcal{P}_i$ responsible to the transaction Φ_i, where each $p \in \mathcal{P}_i$ are typically the owner of each $a \in \mathcal{A}_{in,i}$. To this end, here we suppose that the mapping $\mathcal{A}_{in,i} \overset{\Phi_i}{\longmapsto} \mathcal{A}_{out,i}$ done by $\mathcal{P}_i$ is represented by the data part C_i defined as the following concatenated data object.

$$C_i \triangleq R_i || E_i || ID_{\mathcal{P}_i}. \tag{1}$$

The first component R_i of (1) is called a concatenated list of pointers to data parts C_j's $(j < i)$ of existing transactions T_j's in BBc-1 that have previously generated $\mathcal{A}_{in,i}$. In particular, R_i can be expressed by

$$R_i \triangleq \left[H(C_j) : \mathcal{A}_{out,j} \cap \mathcal{A}_{in,i} \neq \{\}, j < i \right], \tag{2}$$

where $H()$ is a cryptographic hash function. This implies that hash values $H(C_j)$'s are regarded as pointers to C_j's, and hence $H(C_j)$ is also called a *reference* to C_j.

The second and third components E_i and $ID_{\mathcal{P}_i}$ of (1) are an *event description* and a list of identities for $\mathcal{P}_i$, respectively. In particular, E_i describes the transaction Φ_i yielding $\mathcal{A}_{out,i}$ from $\mathcal{A}_{in,i}$ in the following form.

$$E_i \triangleq \left[\text{description of event yielding } a \text{ from } \mathcal{A}_{in,i} : a \in \mathcal{A}_{out,i} \right],$$

where an event is defined as an operation generating an asset $a \in \mathcal{A}_{out,i}$ from $\mathcal{A}_{in,i}$. On the other hand, $ID_{\mathcal{P}_i}$ is the identities of participants responsible to the transaction Φ_i, defined as

$$ID_{\mathcal{P}_i} \triangleq \left[ID_p : p \in \mathcal{P}_i\right],$$

where ID_p represents the identity of p. In other words, $ID_{\mathcal{P}_i}$ enumerates the subjective entities who have actually executed the event E_i and generated $\mathcal{A}_{out,i}$ from $\mathcal{A}_{in,i}$.

We should note that the event description E_i includes any type of description on Φ_i generating $\mathcal{A}_{out,i}$ and/or assets themselves (or their pointer) in $\mathcal{A}_{out,i}$ themselves. Consider the case where Φ_i is a type of non-reproducible operation only with simple description, e.g., the case where $\mathcal{A}_{in,i}$ consists of binary files and Φ_i is their update. Then, objects in $\mathcal{A}_{out,i}$ or pointers to them must be included in E_i to correctly express the output of the transaction Φ_i. In other words, E_i is generated in such a way that the set of assets $\mathcal{A}_{out,i}$ is correctly obtained from itself or by tracking the sequence $C_1, C_2, \ldots, C_i$ with no extra information.

Remark 1. As obviously shown in (2), we include the pointers to previous transactions T_j's instead of assets in $\mathcal{A}_{in,i}$ themselves. This enables us to easily trace back transactions related to the assets directly from C_i. □

Example 1. Here we give a simple example. Let X be only a participant to the i-th transaction, and suppose X updates a PDF data `x.pdf` registered to a `BBc-1` system as an asset. Then, we set $\mathcal{P}_i = \{X\}$, and $\mathcal{A}_{in,i}$ and $\mathcal{A}_{out,i}$ are respectively given by

$$\mathcal{A}_{in,i} = \{\texttt{x.pdf}\,(\text{old})\}, \text{and } \mathcal{A}_{out,i} = \{\texttt{x.pdf}\,(\text{new})\},$$

where the file name `x.pdf` represents the file object itself and we have written old/new to represent the original and update data for the sake of simplicity. Suppose `x.pdf` is registered (or updated) at the j-th transaction ($j < i$), i.e.,

$$\mathcal{A}_{out,j} \cap \mathcal{A}_{in,i} = \{\texttt{x.pdf}\,(\text{old})\}.$$

We then have the data part C_i of the transaction data object T_i as

$$C_i = H(C_j)|||[\text{updated: H}(\texttt{x.pdf}\,(\text{new}))]|||ID_X,$$

where we have assumed that the hash value of `x.pdf` is registered in the system. □

2.1.2 Generation of Signature Part S_i

The aim of the signature part S_i is to prove that the transaction $\mathcal{A}_{in,i} \overset{\Phi_i}{\longmapsto} \mathcal{A}_{out,i}$ is computed by the participants. To this end, participants in $\mathcal{P}_i$ responsible to the transaction Φ_i generates signatures for C_i as follows.

$$S_i \triangleq \left[Sig_p(H(C_i)) : p \in \mathcal{P}_i\right], \tag{3}$$

where $Sig_p()$ is a signature under p's private key.

Example 2. Suppose that as Example 1, the owner of `x.pdf` is X itself. Then we set $\mathcal{P}_i = \{X\}$ and obtain the transaction data T_i as

$$T_i = \underbrace{H(C_j)|||[\text{updated: } H(\texttt{x.pdf}\,(\text{new}))]|||ID_X}_{=C_i} |||[Sig_X(H(C_i))].$$

□

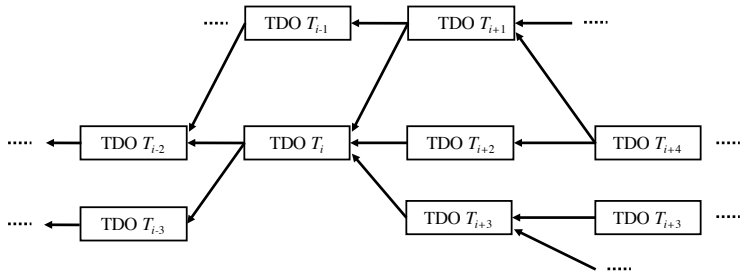

Figure 1: Intuitive image of relationship among BBc-1 transaction data objects (referred to as TDOs), where arrow lines represent references to other transactions

Remark 2. In BBc-1, taking signatures S_i from $\forall p \in \mathcal{P}_i$ can be viewed as obtaining the consensus to the transaction Φ_i. We thus see that the consensus mechanism in BBc-1 is just to request and retrieve participants' signatures via a certain peer-to-peer communication between the registrant generating T_i and each of participants. Unlike Blockchain, no external entity join the consensus phase of BBc-1. □

2.2 Graph-Structured Relationship among Transaction Data

As we described in the previous subsection, a transaction data object T_i has one ore more pointers to existing ones in its data part C_i. This implies that the relationship among transaction data objects can be viewed as a directed graph in terms of reference as intuitively shown in Figure 1. Here we see the directed graph contains no loop since the references are taken only to chronologically-previous transaction data objects. Note that if every transaction data object takes only one reference, the relationship can be viewed as simple (hash) chains.

2.3 Including Cross-References in Transaction Data

BBc-1 has an optional feature to connect a directed graph (see Section 2.2) of transaction data objects with other ones, which is called *cross-references*. In particular, a transaction data object of BBc-1 can be generated so as to contain hashed data parts of transaction data objects registered in external graph. The main aim of cross-reference is to strengthen the tolerance of BBc-1 against impersonation of legitimate participants to widen the range of references in addition to R_i in (2).

In the system enabling the cross-reference, the transaction data object T_i is generated as follows. First, suppose that another external transaction data T_m ($m < i$) would be referred to by T_i as its cross-reference. We then have the data part C_i in the following form with the hash of T_m.

$$C_i = R_i||E_i||ID_{\mathcal{P}_i}||H(T_m). \tag{4}$$

We will give an analysis of this sophisticated option of BBc-1 in Section 3.5.

3 Security Analysis

Here we present a simple security analysis on BBc-1, and clarify its advantages and disadvantages in terms of security.

3.1 Security against Attacks against Transaction Data Objects

First we analyze the structure of transaction data objects from the perspective of attacks against data objects themselves. This subsection does not consider the cross-reference presented in Section 2.3, and the power of the cross reference will be analyzed in Section 3.5.

As given in Sections 2.1 and 2.2, they can be viewed just as hash-connected objects with multiple participants signatures. Here we assume that attackers come from outside of the BBc-1 system, and that all participants of BBc-1 flawlessly work with the defined signature and hash-connecting mechanisms. Suppose that malicious attackers impersonate legitimate participants of the system, and that they want to alter/overwrite a part of the graph or change the order of objects in the graph. We then see that they need to generate correct hash values referred by other parts of the directed graph and compute signatures of legitimate participants. This implies that such security for transaction data objects itself just depends on the adopted hash and signature algorithms.

Next we consider a case where a participant(s) maliciously behaves about transaction data objects. Assume that the attacker(s) tries to remove all successive data objects from a certain point in the graph, and connects new fake data objects with the point instead of removed objects. Recall that basically in BBc-1, any transaction data objects are redundantly stored in multiple BBc-1 node, or else every node stores all the transaction data objects of the system (in small cases). This first implies that such malicious removal of the data objects is never possible except the case where all participants betray and collude with one another. On the other hand, this also means that the attacker is allowed to append new transaction data objects to any point of the directed graph since it is a legitimate participant. Here we recall that BBc-1 requires signatures of participants who responsible to the transaction and assets in the generation of transaction data objects as in Section 2.1. Hence if the attacker wants to maliciously append transaction data objects corresponding to other participants' assets, it needs to generate fake signatures of them. We thus conclude that this type of attacker is allowed to append data objects corresponding only to its asset itself, and that it has no undesirable effect in the system.

As shown in the above, the transaction data objects themselves and the hash-based connection are securely protected from attackers if adopted cryptographic algorithms are secure. In the following subsections, we will give analyses on attacks against the system of BBc-1.

3.2 Sybil Attack

Next we consider the security of BBc-1 system against Sybil attack, i.e., the case where there exist numbers of malicious participants colluding with each other. Considering the 'random' pick-up of signers $\mathcal{P}_i$ in the consensus phase, BBc-1 has the following vulnerability against Sybil attack. In this case, no legitimate participant might be included in $\mathcal{P}_i$, and then attackers can collude and append fake signatures that cannot be correctly verified later. This means we cannot guarantee the integrity of registered data. Also, fake assets and fake transactions might be registered with attackers' signatures.

Protection from this type of attack is out-of-focus of BBc-1, and we obviously see that these can be avoided by applying protection mechanisms in overlaid applications of BBc-1. For example, applications can restricts users to register their underlying BBc-1 system using their external certificates, and applications can be designed so as to choose signers with authenticating signers in the consensus phase. We should note that even in this case, attackers cannot impersonate legitimate participants as long as their private (signature) keys are not leaked to attackers.

3.3 Denial-of-Service Attack

Denial-of-Service (DoS) attack is a classical problem of computer systems, and BBc-1 may suffer from the attack if its overlaid application has no countermeasure against the attack. One possible scenario of DoS to BBc-1 is that malicious attackers simultaneously register huge numbers of transaction data within a short period. Then directed graphs could be incredibly lengthened/expanded and the data size of BBc-1 system grows at unusually-rapid speed. This could finally make the system inaccessible due to huge consumption of storage and network bandwidth. We thus see that applications should be designed in such a way that the overall system is protected from this type of DoS attack. For example for a closed system, we can easily avoid this problem by authenticating registrants at the application layer. For an open system, we can also limit the number of registration of transaction data per a certain unit time.

3.4 Fault Tolerance

We see that in a similar manner to the Blockchain-like mechanism, BBc-1 is basically secure against destruction of data since transaction data objects are redundantly distributed among participant nodes. This implies that transaction data can be correctly queried and is available even if some of participant nodes are not accessible from users. On the other hand, consider the phase to update of a directed graph in BBc-1, i.e., append a new transaction data object to the graph. Here we recall that $\mathcal{P}_i$ is a set of participants responsible to the transaction itself and typically a set of owners of assets in $\mathcal{A}_{in,i}$ and $\mathcal{A}_{out,i}$. We thus see that if an entity in the participant set $\mathcal{P}_i$ is inaccessible, we cannot complete the transaction data object due to the lack of signatures in the signature part S_i in (3), and the directed graph is not updated. From this observation, the overlaid application of BBc-1 must be designed so as to guarantee the accessibility to legitimate participants in $\mathcal{P}_i$ in the generation of a transaction data object. Otherwise in a relaxed security regime, the overlaid application should approve only transaction data objects containing more valid signatures than a certain threshold.

3.5 Power of Cross-Reference

Here we briefly analyze the optional feature of BBc-1, i.e., *cross-reference* defined in Section 2.3.

As mentioned in Section 2.3, the first aim of the cross-reference is to enhance the tolerance of BBc-1 against malicious alternation of transaction data objects in directed graphs. This is obviously guaranteed since the reference in a transaction data object is not only R_i but also $H(T_m)$ as in (4). Even if an attacker successfully altered whole directed graphs including T_m, the malicious behaviour can be immediately detected by verifying hash values included in T_i. Namely, the attacker has to successfully forge multiple graphs in order to achieve its purpose. Consider the case where multiple

independent organizations employ distinct BBc-1 systems and mutually connect them via cross-references through secure API exposing some hash values of transaction data objects. We then see that it is unrealistic for an attacker to compromise one BBc-1 system of a company since the attacker must also compromise systems of all other companies.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, Nov. 2008.

[2] K. Saito and T. Kubo, "BBc-1: Beyond blockchain one." https://beyond-blockchain.org/public/bbc1-design-paper.pdf, Oct. 2017.