About blockchain interoperability

Pascal Lafourcade, Marius Lombard-Platet

Version of Record: https://www.sciencedirect.com/science/article/pii/S0020019020300636
Manuscript_8c3cfae37aebcd72d8d96b472e2353ca

About Blockchain Interoperability

Pascal Lafourcade[a], Marius Lombard-Platet[b]

[a] *Université Clermont-Auvergne, LIMOS CNRS UMR 6158, Aubière, France*
pascal.lafourcade@limos.fr
[b] *Be-Studys, Geneva, Switzerland*
Département d'informatique
de l'ENS, École normale supérieure, CNRS, PSL Research University, Paris, France
marius.lombard-platet@ens.fr

Abstract

A blockchain is designed to be a self-sufficient decentralised ledger: a peer verifying the validity of past transactions only needs to download the blockchain (the ledger) and nothing else. However, it might be of interest to make two different blockchains interoperable, i.e., to allow one to transmit information from one blockchain to another blockchain. In this paper, we give a formalisation of this problem, and we prove that blockchain interoperability is impossible according to the classical definition of a blockchain. Under a weaker definition of blockchain, we demonstrate that two blockchains are interoperable is equivalent to creating a '2-in-1' blockchain containing both ledgers, thus limiting the theoretical interest of making interoperable blockchains in the first place. We also observe that all practical existing interoperable blockchain frameworks work indeed by exchanging already created tokens between two blockchains and not by offering the possibility to transfer tokens from one blockchain to another one, which implies a modification of the balance of total created tokens on both blockchains. It confirms that having interoperability is only possible by creating a '2-in-1' blockchain containing both ledgers.

Keywords: Decentralized ledger, interoperability.

1. Introduction

Blockchain was first introduced in 2008 by Nakamoto in [1]. In their paper, the anonymous author(s) described the first decentralised ledger: a database in which anyone can write, and that is not controlled by a single or a conglomerate of identities. Since then, many other blockchains have been described: Ethereum [2], Ripple [3] and many others. In May 2019, 248 active blockchains were listed on [4].

While many different blockchains exist, there is no direct way of reaching interoperability, at least without a trusted third party. Consider for instance a client willing to convert their Bitcoins to Ether: they would need to consume the amount of Bitcoins they wants to convert and to generate the equivalent amount of Ether. While Bitcoin consumption may be reachable (by sending coins to a non-existing address, such as the address 0), it is impossible to spontaneously generate Ether (or any other

kind of cryptocurrency). For now, the problem is solved with the help of trusted brokers (also called *escrows*), even though other solutions are on their way [5, 6].

The issue of interoperability is solved in some cases, like "atomic exchanges" and hash-locking [7], in which game theory ensures that a broker only benefits when following the protocol. However the question of trustless interoperability in the general context remains open.

Contributions. We introduce a theoretical background to blockchain interoperability, providing a formal definition of a blockchain and of interoperability. We then prove that, by definition, interoperability between two public blockchains is impossible. However, we contend that there may be special conditions under which two blockchains can be interoperable. This leads us to prove the equivalence between two interoperable public blockchains and a ledger emulating both blockchains on two separate registries.

Related Work. The concept of sidechains (a sidechain is a blockchain attached to another blockchain, with exchanges possible between the two blockchains) has been explored in [8]. The authors describe a two-way peg in which a sidechain is fed with an SPV proof, a short proof of the transaction allowing for lightweight clients. The sidechain plays the role of a lightweight client, and can thus allow subsequent operations following the SPV proof. However, this pegging system requires a contest period, during which it is assumed that people will verify that the SPV proof does not come from a fork. Hence, additional trust is required in this model. In a paper from 2016 [7], Buterin lists ways of reaching interoperability, and focuses on trusted inter-chains exchanges, where one sends money on blockchain A and receives some in blockchain B.

Similarly, the Interledger protocol [6] (ILP) allows one to automatize money transfers while leveraging the risk of fraud, thanks to micro-transactions. Yet, ILP is more about escrow synchronization than interoperability as we define it later on. In an ILP transaction from blockchain $\mathcal{A}$ to blockchain $\mathcal{B}$, one must find an escrow having enough money on $\mathcal{B}$ (or several escrows having in total enough money), so the transfer can occur. More generally, we consider that interoperability can for instance allow money to 'disappear' from $\mathcal{A}$ and to 'reappear' on $\mathcal{B}$, without the need for trusted escrows.

Interoperability has been notably implemented in the blockchain network Kadena [9], in which transfers from one blockchain of the network to another is possible. The money is destroyed on one side and generated on the other. Kadena also uses smart contracts for securing escrow transfer. However, there is no indication that Kadena can operate with chains outside of their specific network. So in our terminology, we say that Kadena is a "N-in-1 blockchain", which is to say one blockchain, with several ledgers.

To the best of our knowledge, no theoretical work on interoperability has been done to date. Our work, rather than giving a practical implementation of an interoperable blockchain, gives a theoretical background to the topic, and explores the conceptual meaning of having interoperable blockchains.

Outline. In the next section, we formally define a blockchain and interoperability. In Section 3, we prove that it is impossible by design to have interoperability between blockchains. In Section 4, we show that interoperability is possible with a

weaker definition of the blockchain. Before concluding, in Section 5, we prove that interoperability is equivalent to having a blockchain with two ledgers.

2. Preliminaries

Sets and tuples are noted in calligraphic font: $\mathcal{A}$, algorithms in serif: Mine. When a deterministic algorithm, say $\mathsf{Algorithm}$, returns some value x from some input i, we use the notation $x \leftarrow \mathsf{Algorithm}(i)$. If $\mathsf{Algorithm}$ is randomised, we use the notation $x \xleftarrow{\$} \mathsf{Algorithm}(i)$. A list of elements $e_1,...,e_n$ (in this order) is represented by $[e_1,...,e_n]$. We denote concatenation of two lists a and b with $a \| b$. The set of elements belonging to $\mathcal{A}$ but not to $\mathcal{B}$ is noted $\mathcal{A} \backslash \mathcal{B}$ (this set is also called the difference of $\mathcal{A}$ and $\mathcal{B}$.)

2.1. Blockchain Definition

Various definitions of blockchain have already been given [10, 11]. In this work, we rather give a formalization of blockchains, which we believe is easier to use for proving theoretical results such as the one in this paper.

Intuitively, a blockchain is a chain of transactions. More precisely, each element of the chain (each block) contains several transactions (or one or none), as well a proof needed for consensus to take place. For instance in Bitcoin [1] or similar Proof of Work blockchains, the proof is a nonce (a random number such that the hash of the block is below a threshold value); in a Proof-of-Stake such as the Casper version for Ethereum [2] the proof consists of the successive bets on what the next block will be; in a Proof-of-Elapsed-Time as designed by Intel [12], the proof is instead a certificate obtained from the SGX (a trusted enclave). Note that it exists blockchains not requiring proofs (for instance, one can argue that PBFT consensus does not require proof), in which case we consider the proof is empty.

Definition 1 (Blockchain). Let $\mathcal{T}$ be a set of transactions and $\mathcal{P}$ be a set of proofs. A blockchain is a tuple of elements $\mathcal{B} = (\mathcal{L}, \mathcal{W}, \mathsf{Emit}, \mathsf{Mine})$, where:

- A ledger $\mathcal{L}$ is a list of transactions with their proofs defined by: $\mathcal{L} = [([t_{1,1},t_{1,2},...],p_1),...,[([t_{n,1},t_{n,2},...],p_n)]$ with $t_{i,j} \in \mathcal{T}$ and $p_i \in \mathcal{P}$.
- $\mathcal{W}$ is such that $\mathcal{W} \subset \mathcal{T}$, $\mathcal{W}$ is called the pool of waiting transactions.
- Emit is a deterministic algorithm taking one transaction $t \in \mathcal{T}$ and $\mathcal{W}$ as input, and returning an updated pool $\mathsf{Emit}(t,\mathcal{W}) = W \cup t$.
- Mine is an algorithm taking $\mathcal{L},\mathcal{W}$ and returning a new ledger $\mathcal{L}'$, a new pool $\mathcal{W}'$, where for any $\mathcal{W} \subset \mathcal{T}$, and for $(\mathcal{L}',\mathcal{W}') \xleftarrow{\$} \mathsf{Mine}(\mathcal{L},\mathcal{W})$, we have that $\mathcal{L}'$ is of the form $\mathcal{L} \| [(\text{transacs},p)]$, where transacs is a list containing all elements from $\mathcal{W} \backslash \mathcal{W}'$, and $p \in \mathcal{P}$ a proof.

Furthermore, after a call to Emit or Mine, the ledger $\mathcal{L}$ and the waiting pool $\mathcal{W}$ of $\mathcal{B}$ are updated with the values returned by said algorithms. In other words, Mine and Emit are not pure functions [13], as they have side effects on the blockchain.

At this point, transactions are appended (or not) to the blockchain after a call to Mine. We hereby give a formal definition of what a valid transaction is.

Definition 2 (Valid Transaction). Let $\mathcal{B} = (\mathcal{L}, \mathcal{W}, \mathsf{Emit}, \mathsf{Mine})$ be a blockchain, and let t be a transaction ($t \in \mathcal{W}$), t is a valid transaction for $\mathcal{B}$ (currently in state $\mathcal{L}$) if and only if there exists a block in the ledger returned by Mine containing t.

As we can see, the validity of a transaction depends on the state of the ledger; if a transaction is valid at one point, it may not be valid forever, and reciprocally. For instance, a transaction from user U to user V is valid only as long as U has enough funds. Yet, after the emission and the insertion of the transaction in the blockchain, U may issue other transactions, emptying their wallet. This is the classical issue of double spending.

The same is true for smart contracts: here, they are seen as a special subset of transactions, and they affect the state of the ledger. Because Mine has access to the whole ledger, it can take into account the smart contract's side effects.

Note that Mine is a randomized algorithm, and as such, there is no guarantee that all users will agree on the same ledger. Because blockchain is a decentralised ledger, state synchronisation must be ensured. For this, we introduce a synchronisation algorithm, called Consensus.

Definition 3 (Decentralised Blockchain). A decentralised blockchain is a tuple $\mathcal{B}' = (\mathcal{L}, \mathcal{W}, \mathsf{Emit}, \mathsf{Mine}, \mathsf{Consensus})$ where:

- $\mathcal{B} = (\mathcal{L}, \mathcal{W}, \mathsf{Emit}, \mathsf{Mine})$ is a blockchain,
- Consensus is a deterministic algorithm, taking as input $\mathcal{B}$, a set $\mathcal{S}$ of tuples $(\mathcal{L}_i, \mathcal{W}_i)$ such that $\forall (\mathcal{L}_i, \mathcal{W}_i) \in \mathcal{S}$, we have that $(\mathcal{L}_i, \mathcal{W}_i, \mathsf{Emit}, \mathsf{Mine})$ is a blockchain. Furthermore, for $(\mathcal{L}^*, \mathcal{W}^*) \leftarrow \mathsf{Consensus}(\mathcal{B}, \mathcal{S})$, then $(\mathcal{L}^*, \mathcal{W}^*) \in \mathcal{S} \cup (\mathcal{L}, \mathcal{W})$. In other words, from a list of potential new blocks, Consensus chooses (or accepts) one of them, or rejects them all (and returns $(\mathcal{L}, \mathcal{W})$).
- After a call to Consensus, $\mathcal{B}'$'s ledger and waiting pool components are replaced with the values returned by said algorithms.

The idea of Consensus is that when a peer updates their local version of the blockchain, they first receive possibly more than one new version (i.e., new blocks) from peers. However only one of these new blocks will be accepted, and all the network must agree on this block.

Definition 4 (Secure Blockchain). We say that a decentralised blockchain $(\mathcal{L}, \mathcal{W}, \mathsf{Emit}, \mathsf{Mine}, \mathsf{Consensus})$ is *secure* if it is computationally hard for a user to craft a new ledger $\mathcal{L}'$ and a new transaction pool $\mathcal{W}'$ such that for all S such that $(\mathcal{L}', \mathcal{W}') \in \mathcal{S}$, we have both that $\mathsf{Consensus}(\mathcal{B}, \mathcal{S}) = (\mathcal{L}', \mathcal{W}')$ and $\mathcal{L}$ is not a prefix of $\mathcal{L}'$.

This definition makes a blockchain immune against history rewriting (and double spending), as it is computationally hard to rewrite old blocks.

2.2. Interoperability Definition

The concept of interoperability is to enable two blockchains to work together. A classic blockchain $\mathcal{A}$ accepts transactions because given the current state of $\mathcal{A}$'s ledger, the transaction does not violate $\mathcal{A}$'s rules. Similarly, we say that a blockchain $\mathcal{A}$ that is interoperable with blockchain $\mathcal{B}$ accepts transactions because, given the current

state of $\mathcal{A}$ and $\mathcal{B}$'s ledgers, the transaction does not violate $\mathcal{A}$'s rules. Furthermore, if the rules for said transaction only imply conditions on $\mathcal{A}$'s ledger, then the transaction does not require $\mathcal{B}$ to be valid, and as such does not make use of the interoperability. So an interoperable transaction on $\mathcal{A}$ must be dependent on $\mathcal{B}$'s ledger: if $\mathcal{B}$'s ledger is equal to some values, then the transaction is valid; otherwise it is invalid.

We now give a formalization of this definition.

Definition 5 (Blockchain Interoperability). Let $\mathcal{A}=(\mathcal{L}_A,\mathcal{W}_A,\mathsf{Emit}_A,$ $\mathsf{Mine}_A,\ \mathsf{Consensus}_A)$ and $\mathcal{B}=(\mathcal{L}_B,\mathcal{W}_B,\mathsf{Emit}_B,\mathsf{Mine}_B,\mathsf{Consensus}_B)$ be two decentralised blockchains. Let Ω_A (resp. Ω_B) be the set of all possible values for $\mathcal{A}$'s ledger $\mathcal{L}_A$ (resp. $\mathcal{L}_B$). $\mathcal{A}$ is *interoperable* with $\mathcal{B}$ if there exists:

- a transaction $t\in\mathcal{T}$,
- a non-empty subset $\omega_A\subset\Omega_A$,
- a non-empty proper subset $\omega_B\subsetneq\Omega_B$

such that there exists a block containing t that is accepted by $\mathsf{Consensus}_A$ if $\mathcal{L}_A\times\mathcal{L}_B\in\omega_A\times\omega_B$, and rejected otherwise.

$\mathcal{A}$ and $\mathcal{B}$ are *interoperable* if they are both interoperable with each other.

3. General Impossibility of Interoperability

Our first result is to show that it is impossible to have interoperability between two blockchains in general.

Theorem 1 *Under the definitions 3 and 5, blockchain interoperability is impossible.*

Proof. Assume that an interoperable transaction t exists. Then there is a set ω_B of possible ledger values of $\mathcal{B}$ for which a block containing t is accepted by $\mathsf{Consensus}_A$, if $\mathcal{L}_B\in\omega_B$. Moreover, if $\mathcal{L}_B\in\Omega_B\backslash\omega_B$, then $\mathsf{Consensus}_A$ will refuse any block containing t.

However, $\mathsf{Consensus}_A$ only takes $\mathcal{A},S$ as arguments, where S is a set of tuples $(\mathcal{L}_i,\mathcal{W}_i)$ (see Definition 3). As a consequence, $\mathsf{Consensus}_A$ is independent from $\mathcal{B}$, and especially from $\mathcal{L}_B$. Then, if t is accepted by $\mathsf{Consensus}_A$ when $\mathcal{L}_B\in\omega_B$, then t is also accepted by $\mathsf{Consensus}_A$ when $\mathcal{L}_B\in\Omega_B\backslash\omega_B$; this implies that $\Omega_B\backslash\omega_B=\emptyset$, i.e., $\omega_B=\Omega_B$, which is a contradiction with the hypothesis of Definition 5, namely that ω_B is a proper subset of Ω_B. □

This result is actually quite straightforward if we remember that a blockchain is, by construction, made to be self-sufficient: no blockchain can rely on external data. Especially, no blockchain can rely on another blockchain for asserting the validity of a transaction. Hence, interoperability is a contradiction of one of the intrinsic characteristics of blockchain.

The interpretation of the result is as follows: without additional assumptions, interoperability between two blockchains is impossible. Therefore, to achieve interoperability further assumptions need to be made. For instance, in the two-way pegged blockchain mechanism, a dispute period is required for each interoperabilty operation; as the blockchain cannot know by itself whether the proposed SPV proof is the one of the latest block.

4. Interoperability with a Weaker Definition

Even though blockchain is not suited for interoperability *stricto sensu*, we can generalise our blockchain definition, in order to make a blockchain interoperable.

Hypothesis 1 *We assume that for two blockchains* $\mathcal{A} = (\mathcal{L}_A, \mathcal{W}_A, \mathsf{Emit}_A, \mathsf{Mine}_A, \mathsf{Consensus}_A)$ *and* $\mathcal{B}$*, with* $\mathcal{A}$ *both* Mine_A *and* $\mathsf{Consensus}_A$ *have access to both* $\mathcal{A}$ *and* $\mathcal{B}$*:* $\mathsf{Consensus}_A$ *is of the form* $\mathsf{Consensus}_A(\mathcal{A}, \mathcal{B}, \mathcal{S})$*, and* Mine_A *is of the form* $\mathsf{Mine}_A(\mathcal{L}_A, \mathcal{L}_B, \mathcal{W}_A, \mathcal{W}_B)$.

We now use the notation $\mathsf{Consensus}_A(\mathcal{A}, \mathcal{B}, \cdot)$ to note the new consensus algorithm. Hence, the 'version' of $\mathsf{Consensus}$ in the previous definition, is now noted $\mathsf{Consensus}_A(\mathcal{A}, \emptyset, \cdot)$. Similarly, the non-interoperable version of Mine is now noted as $\mathsf{Mine}_A(\mathcal{L}_A, \emptyset, \mathcal{W}_A, \emptyset)$.

Definition 6 (Interoperable transaction). Under the assumption hypothesis 1, a transaction t on the blockchain $\mathcal{A}$ is said to be interoperable with $\mathcal{B}$ if t can be accepted by $\mathsf{Consensus}_A(\mathcal{A}, \mathcal{B}, \cdot)$ but cannot be accepted by $\mathsf{Consensus}_A(\mathcal{A}, \emptyset, \cdot)$.

In this context, we have the following result.

Theorem 2 *Under Hypothesis 1, it is possible to build interoperable blockchains.*

Proof. Note that we already know that interoperable blockchains exist, such as Kadena [9] or other blockchains listed in [5], but we give an example of interoperable blockchain in our own theoretical framework.

Consider two decentralised blockchains $\mathcal{A} = (\mathcal{L}_A, \mathcal{W}_A, \mathsf{Emit}_A, \mathsf{Mine}_A, \mathsf{Consensus}_A)$ and $\mathcal{B}$. On the blockchains we define *accounts*. An account ownership is defined by the knowledge of a private key, and for simplicity the public key is assimilated to the account itself. A transaction $t \in \mathcal{T}_A$ (resp. $\mathcal{T}_B$) specifies the sender's public key, the receiver's public key, an amount and a signature of the previous fields by the sender's private key. An account i on blockchain $\mathcal{A}$ (resp. $\mathcal{B}$) is designed by i_A (resp i_B).

We build blockchain $\mathcal{B}$ so that it is interoperable with blockchain $\mathcal{A}$ in the following sense: a user can 'create' money on $\mathcal{B}$ if and only if at least the same amount of money has been consumed on $\mathcal{A}$, by sending it to a 'bin' account.

We first note that accounts owned by nobody exist. In our scheme, in most cryptosystems the public key 0 (consisting of only zeroes) is not linked to any private key. Thus, while the account 0_A exists and money can be transferred on this account, it cannot be claimed by anyone.

Let us construct B in order to fulfil the previous requirements. First, let us define the interoperability transactions $t^*(m_B, pk_B)$, which sends some amount of money m_B from 0_B to the account pk_B on $\mathcal{B}$. $t^*(m_B, pk_B)$ is only valid if[1] there is at least one transaction on $\mathcal{L}_A$ sending m to the account 0_A, with $m > m_B$.

[1] Note that for a real cryptocurrency more checks would be needed for any practical use, notably because of the fact that in the current setting, anyone can withdraw m from 0_B as many times as they want. However, for the sake of simplicity, we only describe a simple, naive interoperability operation here, so these checks are omitted.

Then, let us construct Mine_B: a transaction t is valid for $\mathsf{Mine}_B(\mathcal{L}_B,\mathcal{L}_A,\mathcal{W}_B,\mathcal{W}_A)$ if and only if t is valid for $\mathsf{Mine}_\mathsf{A}(\mathcal{L}_A,\emptyset,\mathcal{W}_A,\emptyset)$, or if both statements are true:

- t is an interoperability transaction transferring some amount of money m from 0_B to an account on $\mathcal{B}$,
- $\mathcal{L}_A$ contains a transaction sending at least m on the zero-address 0_A.

Similarly, $\mathsf{Consensus}_B(\mathcal{B},\mathcal{A},\cdot)$ is conceived to accept new ledgers that would have been accepted by $\mathsf{Consensus}_A(\mathcal{B},\emptyset,\cdot)$, as well as ledgers where the new blocks are constituted solely of transactions that are accepted by $\mathsf{Consensus}_A(\mathcal{B},\emptyset,\cdot)$ and valid interoperability transactions (valid in the meaning that at the time of their incorporation in the ledger, the sender has enough funds to emit the transaction).

With this construction, we immediately get that $\mathcal{B}$ is interoperable with $\mathcal{A}$: a user can transfer assets from $\mathcal{A}$ to $\mathcal{B}$, which is shown by the fact that some transactions (here denoted t^*) are only valid on $\mathcal{B}$ if the sender has enough funds on $\mathcal{A}$.

□

5. Equivalence of Interoperable Blockchains with a Single Blockchain

Even though interoperable blockchains can be tweaked into existence, we argue that they are conceptually equivalent to a single blockchain. More precisely, we argue that they are equivalent to one blockchain, composed of two ledgers. Such a blockchain can be easily implemented: if the first bit of the transaction is 0, then apply the transaction to the first ledger, and if 1 to the second.

We say that two blockchains are equivalent if any valid transaction on one blockchain corresponds to one valid transaction on the other blockchain. This definition implies that two equivalent blockchains will have very similar evolutions of their ledgers. As Mine is not deterministic, we cannot ensure that the two ledgers will be identical, but the definition we give is enough for practical uses.

Definition 7 (Blockchain equivalence). Let there be two blockchains $\mathcal{A}=(\mathcal{L}_A, \mathcal{W}_A,\mathsf{Emit}_A,\mathsf{Mine}_A)$ and $\mathcal{B}=(\mathcal{L}_B,\mathcal{W}_B,\mathsf{Emit}_B,\mathsf{Mine}_B)$ accepting transactions from $\mathcal{T}_A$ and $\mathcal{T}_B$, respectively. $\mathcal{A}$ and $\mathcal{B}$ are said to be equivalent if there exists a bijection $\varphi:\mathcal{T}_A\to T_B$ such that, if both ledgers are equivalent, then there is an equivalence of the valid transactions. In mathematical terms, $\varphi(\mathcal{L}_A)=\mathcal{L}_B\Rightarrow\forall t_A\in\mathcal{T}_A,t_A$ is a valid transaction for $\mathcal{A}\Leftrightarrow\varphi(t_A)$ is a valid transaction for $\mathcal{B}$).

Note that $\varphi(\mathcal{L}_A)$ is the generalization of φ to ledgers: if $\mathcal{L}_A = [[t_{1,1},t_{1,2},...]$, $...,[t_{n,1},t_{n,2},...]]$, then $\varphi(\mathcal{L}_A)=[[\varphi(t_{1,1}),\varphi(t_{1,2}),...],...,[\varphi(t_{n,1}),$ $\varphi(t_{n,2}),...]]$, in the case of a ledger without proofs. If the ledger has proofs (see Definition 1), φ would need to work on a projection of the ledger: a projection in which every poof is removed. This subtlety has been removed from the definition for the sake of simplicity.

For instance, let us assume that two blockchains are equivalent, and two smart contracts being the reciprocal image of each other. This means that whatever transaction triggers one smart contract, the effects on the state of the blockchain will be equivalent to the effects on the state of the image blockchain: in both cases, the acceptable

elements after the transaction are the same (up to a bijection). This definition does not guarantee that the smart contract will behave identically: for instance, one could image a smart contract updating a useless write-only variable, which by definition does not affect the set of future acceptable transactions as it is write-only. However, it ensures that the behaviour of the blockchain is strictly the same in both cases.

Theorem 3 *A decentralised blockchain $\mathcal{A}$ interoperable with a blockchain $\mathcal{B}$ is equivalent to a decentralised blockchain $\mathcal{C}$ containing both $\mathcal{A}$ and $\mathcal{B}$'s ledgers.*

PROOF. Let $\mathcal{A} = (\mathcal{L}_A, \mathcal{W}_A, \mathsf{Emit}_A, \mathsf{Mine}_A, \mathsf{Consensus}_A)$ and $\mathcal{B}$ be two decentralised blockchains, with $\mathcal{A}$ being interoperable with $\mathcal{B}$. $\mathcal{A}$ being interoperable with $\mathcal{B}$, we have Mine_A of the form $\mathsf{Mine}_A(\mathcal{L}_A, \mathcal{L}_B, \cdot)$, and $\mathsf{Consensus}_A$ of the form $\mathsf{Consensus}_A(\mathcal{A},\mathcal{B},\cdot)$.

Let $\mathcal{T}_A$ (resp. $\mathcal{T}_B$) be the set of transactions for $\mathcal{A}$ (resp. $\mathcal{B}$). Note that $\mathcal{T}_A$ contains interoperability transactions. Let $\mathcal{C}$ be the tuple $\mathcal{C} = (\mathcal{L}_C, \mathcal{W}_C, \mathsf{Emit}_C, \mathsf{Mine}_C, \mathsf{Consensus}_C)$.

We define the set of transactions for $\mathcal{C}$, $\mathcal{T}_C = (\mathcal{T}_A \times \emptyset) \cup (\emptyset \times \mathcal{T}_B)$. Let c_A (resp. c_B) be the canonical projector of $\mathcal{T}_C$ on $\mathcal{T}_A$ (resp. $\mathcal{T}_B$).

For $(\mathcal{L}_A \| [(transacs_A, p_A)], \mathcal{W}'_A) = \mathsf{Mine}_A(\mathcal{L}_A, \mathcal{L}_B, c_A(\mathcal{W}_C))$ and $(\mathcal{L}_B \| [(transacs_B, p_B)], \mathcal{W}'_B) = \mathsf{Mine}_B(\mathcal{L}_B, c_B(\mathcal{W}_C))$, we define: $\mathsf{Mine}_C(\mathcal{L}_C, \mathcal{W}_C) = (\mathcal{L}_C \| [(transacs_A \times \emptyset \| \emptyset \times transacs_B, p_A \times p_B), (\mathcal{W}'_A \times \emptyset) \cup (\emptyset \times \mathcal{W}'_B)])$

Simply put, Mine_C is a parallelisation of Mine_A and Mine_B: a block proposed by Mine_C is a block comprised of the transactions accepted by Mine_A and Mine_B.

Similarly, $\mathsf{Consensus}_C$ is built as a parallelisation of $\mathsf{Consensus}_A$ and $\mathsf{Consensus}_B$. If $\mathsf{Consensus}_A(\mathcal{A}, \mathcal{B}, c_A(\mathcal{S}_C)) = (\mathcal{L}^*_A, \mathcal{W}^*_A)$ and $\mathsf{Consensus}_B(\mathcal{B}, c_B(\mathcal{S}_B)) = (\mathcal{L}^*_B, \mathcal{W}^*_B)$, then we define $\mathsf{Consensus}_C = (\mathcal{L}_A \times \emptyset \| \emptyset \times \mathcal{L}_B, \mathcal{W}_A \times \emptyset \cup \emptyset \times \mathcal{W}_B)$.

By construction, we see that $\mathcal{C}$ is a decentralised blockchain. Furthermore, by construction each transaction accepted by Mine_C is either accepted by Mine_A or Mine_B and, conversely, each transaction accepted by Mine_A or Mine_B is accepted by Mine_C, hence the equivalence of the blockchains. □

In practice, Theorem 3 means that creating two interoperable blockchains is equivalent to creating one blockchain, with a ledger divided into two separate registries: a '2-in-1' blockchain. So while creating interoperable blockchains (with a lax definition of a blockchain) is possible, we argue that the conceptual interest of doing so is limited. However, it may be interesting to create an interoperable blockchain on top of an already existing blockchain. Doing so allows both blockchains to fully operate, without the older blockchain being affected by anything. Nonetheless the obvious restriction is that only one of the two blockchains will be interoperable with the other, with all the limits implied by this fact.

6. Conclusion

In this paper, we explored the possibility of making two blockchains interoperable. We showed that, under classical definitions, it is impossible to make a blockchain interact with anything other than itself. If we relax the definition, we do get the possibility of interoperable blockchains, but doing so is equivalent to creating a '2-in-1' blockchain, i.e., a blockchain with two ledgers.

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2009).
URL http://www.bitcoin.org/bitcoin.pdf

[2] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform (2014).
URL https://github.com/ethereum/wiki/wiki/White-Paper

[3] D. Schwartz, N. Youngs, A. Britto, The ripple protocol consensus algorithm (2014).
URL https://ripple.com/files/ripple_consensus_whitepaper.pdf

[4] CryptoID, Crypto-currency blockchain explorers (2019).
URL https://chainz.cryptoid.info/

[5] S. Johnson, P. Robinson, J. Brainard, Sidechains and interoperability, arXiv e-prints (2019). arXiv:1903.04077.

[6] S. Thomas, E. Schwartz, A protocol for interledger payments (2015).
URL https://interledger.org/interledger.pdf

[7] V. Buterin, Chain interoperability (2016).
URL https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf

[8] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, P. Wuille, Enabling blockchain innovations with pegged sidechains (2014).
URL http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains

[9] W. Martino, M. Quaintance, S. Popejoy, Chainweb whitepaper (2018).
URL http://kadena2.novadesign.io/wp-content/uploads/2018/08/chainweb-v15.pdf

[10] J. Garay, A. Kiayias, N. Leonardos, The bitcoin backbone protocol: Analysis and applications, in: Advances in Cryptology - EUROCRYPT, 2015.

[11] A. F. Anta, K. Konwar, C. Georgiou, N. Nicolaou, Formalizing and implementing distributed ledger objects, SIGACT News (2018).

[12] IBM Hyperledger Consortium, Hyperledger sawtooth (2017).
URL https://sawtooth.hyperledger.org/docs/core/releases/latest/index.html

[13] B. Milewski, Pure functions, laziness, i/o, and monads (2014).
URL https://www.schoolofhaskell.com/school/starting-with-haskell/basics-of-haskell/3-pure-functions-laziness-io