Decentralized Financial Intermediation Beyond Blockchains

Fabio Massacci[a,*], Chan Nam Ngo[a], Julian Williams[b]

[a]*Department of Information Engineering and Computer Science, University of Trento, Trento, Italy.*
[b]*Durham University Business School, Durham, UK.*

Abstract

Blockchains and Byzantine Fault Tolerance form the basis of decentralized currencies and ledgers such as Bitcoin, Ripple, ZeroCash, and Ethereum. A large slate of literature has focused on the currency aspects (e.g. anonymity, independence from central banks, etc.). We argue that, as-far-as Distributed Payment Transactions Networks (PTNs) are concerned, there are other, possibly more interesting, properties. This paper provides a systematic review of both traditional PTNs and their analogues in decentralized ledgers and associates different technological features to the corresponding business and financial requirements. We provide a conceptual classification of the key properties (value creation, payment promise, transaction fulfillment, and value preservation). We map existing (distributed) PTNs into the classification showing different alternatives are possible. Furthermore, the ideas behind distributed ledgers can be extended beyond payments and contracts. We illustrate the idea of *derivatives-contracts-as-programs* that are marked to market (or an account that is margined) automatically by computations run on, and whose ownership transitions are recorded, in a distributed payment network.

Keywords: Blockchain, Byzantine Fault Tolerance, Distributed Payment Transaction Network, Smart Contract, Derivative Contracts As Programs

1. Introduction

A traditional classification of payment systems distinguishes between token-based and account-based ones [1]. A good example of the token-based systems is E-cash [19], where parties exchange electronic tokens to represent value, while Visa and MasterCard exemplify account-based systems in which money is stored only as numbers in banking accounts.

Token-based systems, comparing to account-based ones, provide potentially stronger anonymity as it is hard to identify the payer of the token used in a transaction. For instance, the E-cash coin's original owner is untraceable [20] unless the coin is spent twice in off-line transactions[1]. As a result, many previous works focused on the security aspect of the token-based payment systems such as authenticity, integrity, anonymity, etc. [19, 20, 64, 15].

[*]Corresponding Author

Email addresses: `fabio.massacci@unitn.it` (Fabio Massacci), `channam.ngo@unitn.it` (Chan Nam Ngo), `julian.williams@durham.ac.uk` (Julian Williams).

Submitted for publication. Comments welcomed. Cite as F. Massacci, C.N. Ngo and J. Williams, Decentralized Financial Intermediation Beyond Blockchains. Available at SSRN: http://ssrn.com/abstract=2794913

[1]In online E-cash, the double-spent coin will be rejected immediately.

At the same time, one of the limitations of current electronic token-based systems is being centralized, e.g. both payer and payee must open an account in the *same* E-cash *bank* to be able to transfer funds. New decentralized payment systems, e.g. Bitcoin [56], ZeroCoin [54], ZeroCash [8], Ethereum [28], Ripple [62] or RSCoin [24], have been widely adopted as the systems are maintained by *not only a single financial institution* but a distributed set of nodes world-wide[2]. The decentralization of these payment networks relies on the consensus protocols such as Proof-of-Work [56, 67, 30], or the Ripple protocol [66], based on [17].

Systematic Overview. So far, most attention has been devoted to the "currency" and cryptographic aspects of decentralized PTNs (see for example survey [14]) with either enthusiasts (claiming democratization), or vibrant detractors (claiming collusion with money laundering). This article seeks to provide a different viewpoint. To this extent we start by a systematic review of deployed PTNs, whether centralized or distributed, traditional or digital which are dissected into their engineering components. By linking each technological component to a possible business objectives of PTNs it emerges clearly that some choices are not mandatory and several interesting new combinations are possible to produce new systems for the broad variety of financial services. One of the key aspects of our analysis will be an understanding of how a Distributed PTN can be (or in many cases has already been) re-tooled for different tasks.

For example, one of the critical aspects of building a Distributed PTN, e.g. Bitcoin, comparing to a centralized PTN, for instance E-cash, is its implementation of consensus building within a distributed system. A Distributed PTN requires agreement and consistency across the nodes on how transactions are initiated and cleared. If it can be tricked into presenting different transaction records to different parties then (a) parties within the system will be vulnerable to fraud and (b) persistent failures will lead to the discontinuity of the system as the PTN collapses.

These are classical Byzantine fault tolerant (BFT) problems, well studied in distributed system. The transition from impossibility results in general BFT approaches (see [48, 34]), to a practical BFT approach relies heavily on the assumptions of the type and structure of the adversary seeking to subvert the consensus (and hence in our case defraud the ledger) and the effectiveness of cryptographic techniques in facilitating the conveyance of messages between nodes in the distributed system. The applicability of those assumptions to a traditional PTN is what makes possible the transformation of a traditional PTN with a handful of trusted critical third parties to a distributed PTN with less trusted and/or less critical third parties. Many of the current implementations of distributed ledgers (such as Ripple) already implicitly or explicitly utilize the essentials of the practical BFT algorithms in their operation (see [14] for further details).

Decomposition of Features and Threats. To illustrate the decomposition of both the engineering structures and requirements of transaction systems, we provide a summary of how various transaction systems operate, categorize primitive elements, and finally map them into the business requirements of PTNs. Such comparative analysis has an immediate advantage: even a cursory look at the Tables in §4 and Appendix A clearly shows that the very same business requirement can be met in different ways and by involving different actors; slightly different combinations among rows or columns can yield different systems.

Similarly, the Tables in §5 organize the threat mitigating mechanisms of each example payment system in an insightful manner. Our decomposition framework provides useful explanation why

[2] Bitcoin, ZeroCash and Ripple are operational while RSCoin is still a laboratory experiment.

Table 1: Payment Transaction Networks

Among all listed PTNs, E-cash, Bitcoin, ZeroCoin, Ripple, RSCoin will be selected to demonstrate different features in §(4). An interesting application of an account-based, decentralized and programmed-value PTN, the distributed CME implementation will be discussed in §(7.1).

	Fixed	**Programed**
Token	**Centralized:** E-cash **Decentralized:** Bitcoin, ZeroCoin, ZeroCash, RSCoin	**Centralized:** - **Decentralized:** Bitcoin scripts, Ethereum
Account	**Centralized:** VISA, MasterCard **Decentralized:** Ripple	**Centralized:** LexiFi **Decentralized:** A distributed CME implementation [?]

a certain step in a payment system is necessary, either to implement a business objective or to mitigate a PTN threat (or both at the same time). Thus, to further enhance a system, the protocol designers need simply to locate the step's related components and make corresponding changes.

Derivatives-contracts-as-programs. Furthermore, payment systems are not limited to simple transactions, i.e transferring a *fixed-value*, in which the functionalities are simply *deposit*, *transfer* and *withdraw*[3]. Advanced systems allow extended functionalities, i.e. *programmed-value* transactions, in which we replace the constant number in a transaction with a program that upon execution return a *derived* value, e.g. bond dividend, forward and futures contract. Hence, in this paper we further illustrate the idea of *derivatives-contracts-as-programs* that are marked to market (or an account that is margined) automatically by computations run on, and whose ownership transitions are recorded, in a distributed payment network. An illustrative example of such advanced PTN is the futures market of Chicago Mercantile Exchange (CME) [21]. Table 1 gives examples of a three-dimensional classification of current payment systems based on orthogonal criteria, i.e. token/account-based, centralized/decentralized and fixed/programmed-value PTNs.

In addition, programmed-value payment often introduces the non-monotonicity. Typical PTNs (e.g. Bitcoin, ZeroCash) are monotonic: no actions by good guys can make the security commitments of another good guy invalid. No other such protocol exists for digital currencies. Program-value payment, e.g. a futures contract, is however non-monotonic, i.e. a valid offer can invalidate a previously good inventory (see the example in §(7.2)). In this paper we provide some insights into this novel challenge in realizing distributed financial intermediation.

Paper Organization. The remainder of this paper is organized as follows: §(2) provides a review of classic non-distributed payment systems. Additionally, we will introduce the electronic communications network as payment systems and then provide a short overview of the development of these systems. §(3) provides a non-exhaustive overview of the current cryptopayment universe, whilst §(4) presents our decomposition of the features and reviews the critical engineering issues associated with them. §(5) and §(6) provide a summary of the threat environment and how nodes may be used by malicious attackers to subvert, destroy or commit fraud on the network. Finally, while most of the previous sections describe simple fixed-value PTNs, in §(7) we postulate some ideas for the future. Critically, we outline how a distributed ledger might be used in an active

[3]e.g. Bitcoin, ZeroCoin, ZeroCash and Ripple

and automated processing mechanism to replicate more complex markets, we use a futures market as an example, i.e. a distributed CME implementation, the account-based, decentralized and programmed-value PTN in Table 1. §(8) then provides a brief summary.

2. A primer on Traditional Payment Transaction Networks

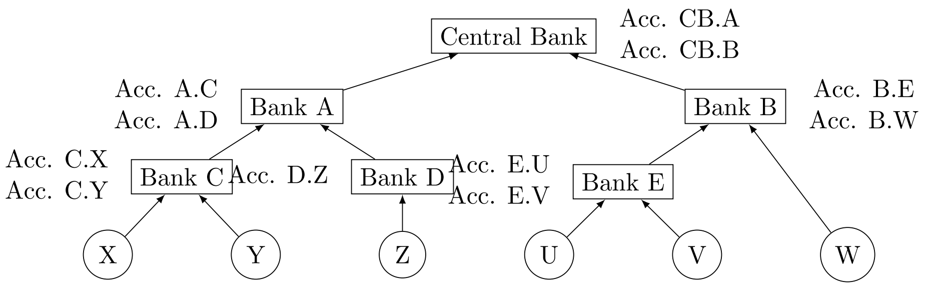
In most PTNs without physical cash transfers, the payer gives the payee a cheque (we use the Anglo- French spelling to avoid confusion with our use of the work "check" in the sense of verification), with which the payee can withdraw cash from the payer's bank or lodge the cheque at the payee's own bank so that the funds eventually are transferred to the payee's account from the payer's. When a payee lodges the cheque, the funds will usually be credited after some time and the cheque will be sent to a clearing house where the payer's bank will validate the cheque. If the cheque is valid, e.g. funds available and signature matched, the payer's bank will debit the amount written on the cheque to the payer's account once the two banks've settled the amount by crediting and debiting the respective accounts at their central bank. An invalid cheque will be returned and the payee will not be able to acquire the funds. In cheque payment, the payee suffers from such reversible payment and usually is on the disadvantage side (e.g. if the payee, usually a merchant, has shipped the goods upon receiving the cheque). Electronic transfer essentially eliminates the physical transfer of cash and their risk of both delayed authentication, e.g. signature matched, and delay authorization, e.g. funds available.

Fig. 1 illustrates the tiered-banking model. A central bank rests on the top of the hierarchy whereas below are multiple levels of dependent banks and the clients at the lowest level. The central bank may also control the monetary supply of the currency and its dependent banks have to hold an account in its ledger. The clients are at the leaves of the hierarchy. In the correspondent-banking model of Fig. 2, the banks A and B form direct relationship with each other, possibly through an exchange, so that they can settle the transactions without a third party. In the case of an exchange, a single bank may have accounts in both currencies and will convert the currencies and the clients suffer from some fees due to currency conversion. Alternatively, both of the payer and payee's banks may have a correspondence account with a *multinational bank* and the exchange is then conducted as an internal transfer by the *multinational bank*[4]. In real world scenarios, the transaction may go through multiple levels and two central banks that have correspondent agreements, thus combining Fig. 1 and Fig. 2. The more third parties the money flows through, the more transaction fees final clients have to pay.

2.1. The Development of Traditional Digital PTNs

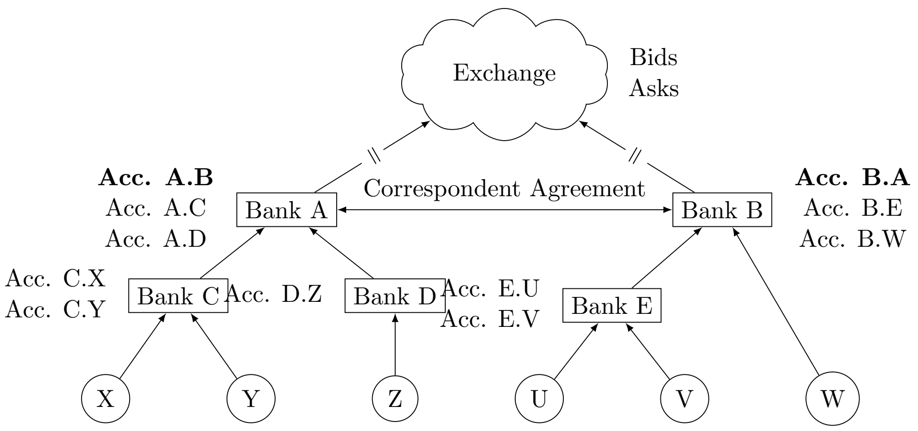
As in [1], traditional electronic payment systems typically either replace the direct transfer from payer to payee, i.e. token-based systems, e.g. E-cash [20] and electronic cheque [3], or indirect form of transfer, i.e account-based, e.g. Debit Cards and Credit Cards. PayPal [58] offers the capability to process payment via the Internet by using the client's cards detail. Recently, innovations in payment technologies provides wrappers to existing payment infrastructures, e.g Google Wallet [36] and Apple Pay [4] allow users to pay with their authenticated phone.

[4]A correspondence account held by a banking institution to receive deposits from, make payments on behalf of, or handle other financial transactions for another financial institution, usually overseas. Commonly, these are referred to as "nostro" and "vostro" accounts in international finance. A "nostro" (ours) account is an account of our money, held by the other bank. A "vostro" (yours) is an account of the other bank money, held by us.



The transactions between X and Y only need Bank C. In the case of the transactions between X and Z, Bank A needs to settle the account A.C and A.D to transfer money from Bank C to Bank D and Bank C will deduce the account C.X while Bank D credit the account D.Z. More sophisticated transaction between Y and V will involve more banks, including the Central Bank, and thus accrue more transaction fees.

Figure 1: The tiered-banking model



Bank A and Bank B hold an account of each other in their ledger, A.B and B.A respectively. The transactions between X and V will need the cooperation of both Bank A and B to settle the amount. Obviously, the settlement also needs to be propagated down to Bank C and Band E. The transactions could also involve currency exchange. In this case, Bank A and Bank B will go through the Bid/Ask orders to sort out the exchange.

Figure 2: The correspondent-banking model

Payments across intermediaries are not necessarily instantaneous. In the *Differed Net-Settlement System*, the bank intermediaries may settle the outstanding differences at the end of the day. A Real-Time Gross-Settlement System (RTGS) provides the function of both “real-time” and “gross-settlement”, which means low-volume, high-value and immediate payments. In presences of securely authenticated parties, RTGS is a low-risk transaction environment as payments are final and irreversible at the execution point. An example of RTGS is CHAPS in the UK where its counterpart is BACS that handles Net-Settlement. Other examples include FedWire and CHIPS in the US. The Eurosystem is in the process of implementing the Target2 system which provides RTGS for banks across the European Union.

For international payments, financial institutions around the globe make use of SWIFT (Society for Worldwide Interbank Financial Telecommunication) to send the payment orders in a secure and standard environment. Then the payments will be settled using their correspondent accounts. SWIFT identifies the financial institutions via a Business Identifier Code (the SWIFT code).

The proposal of “programming languages for financial contracts” (e.g. the Ethereum smart contract [28]) is far from new. It has been first proposed by J. M. Eber in [39] and was marketed by the company LexiFi for the management of traditional financial contracts where the contracts can be expressed and automated (centrally) with Modeling Language for Finance (MLFi) [65].

In parallel to this mostly “monolithic” architecture, the new architectures aim to transfer cash quantities without centralized clearing and we will now give a short technical summary before providing a new set of insights.

3. A Brief History of Crypto PTNs

The concept of a completely digital currency dates back to 1982 when David Chaum introduced the E-cash scheme [20]. E-cash took a “central bank” approach with a completely flat hierarchy where there is no intermediary institution. The scheme also provided such strong anonymity that even cooperation between the E-cash coin issuing bank and the merchant would still fail to identify the spender of the coin. The scheme allowed both online and offline verification of the coin. In the case of offline verification, the payer’s identity could be revealed if the coin is spent twice [20].

Subsequent attempts at a purely digital currency were B-Money [23], hashcash [31], and BitGold [69] which initially utilize “Proof-of-Work” [38], a hard cryptographic computational puzzle, as a mean of mitigating Sybil attacks [25] and eventually determining the inherent value for the medium of exchange (in Bitcoin). These conceptual ideas later matured into the Bitcoin-esque approach which is free of a trusted central bank.

The idea of a public ledger predates Bitcoin but the original idea was not decentralized [64]. The Bitcoin protocol introduces a decentralized transaction database, namely blockchain [56]. The blockchain is shared by all nodes participating in the Bitcoin system. Every executed transaction is included in the blockchain so that any node can track the balance of each address at any point of the system history to later validate a transaction in the network. The transfer of value in the Bitcoin network is carried out via the transfer of amount of BTC (the Bitcoin currency) from an address to another one. The receiver will have to provide the private key to unlock the funds that they received in an earlier transaction. A more thorough review on Bitcoin’s technical details can be found at [70].

Since the launch of Bitcoin in 2009, many cryptocurrencies have been developed. Some use different hash functions than Bitcoin. Dogecoin, Litecoin, and PotCoin utilize the hash function of Scrypt [59] which is more memory-costly to deter the use of hardware-based mining devices. Dash

[26] utilizes X11 which consists of eleven different hashing algorithms in a chain while Primecoin [41] requires finding a prime chain composed of Cunningham chain and Bi-twin chain. BlackCoin [71] and Nxt [57] apply the Proof-of-Stake [13] mechanism where the miners need to prove the ownership of a certain amount of cryptocurrency.

Many of the new cryptocurrencies try to offer extended functionality. At an extreme this is represented by Ethereum which claims a "programming languages for financial contracts".

> "A built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code." [28]

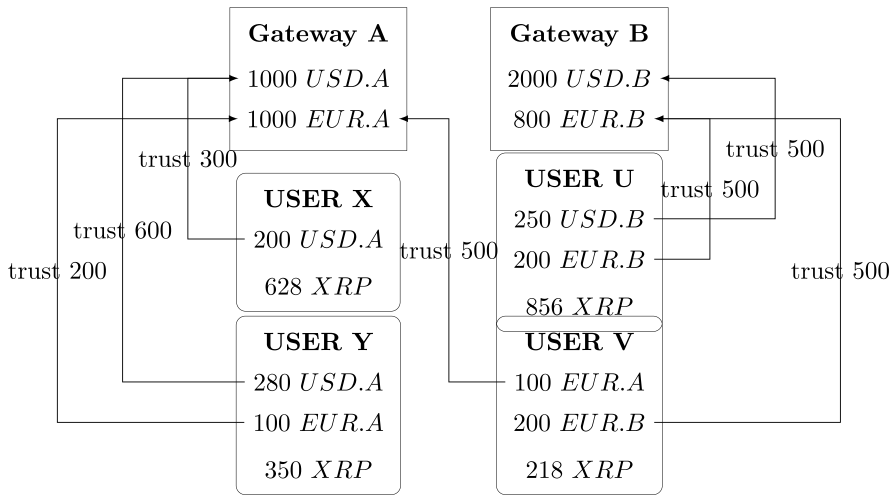
Interestingly, cryptocurrencies can also be used to incentivize correct behavior and fairness in multi-party computation [43] by using crypto tokens in the claim-and-refund transaction flavors where each party makes "conditional" public transactions to each other: the parties that finalize the computation can claim the fund otherwise it is refunded to the original payer [44, 45]. An alternative solution is to commit and lock the deposits before the computation which are only released at the end [42].

In contrast to the preceding types, ZeroCoin [54] is a combination of E-cash scheme and Bitcoin to improve Bitcoin's anonymity. Subsequently, the ZeroCoin apporach was implemented as Moneta. Another case is Monero which offers sender and receiver privacy by hiding connection between the sender and the receiver's addresses and applies Cryptonote [63] which utilizes traceable link signature [33] to confirm the validity of the transactions. ZeroCash [8] offers stronger anonymity comparing to ZeroCoin, as ZeroCoin is only a "coin washing service" while ZeroCash is an actual privacy-preserving payment scheme. ZeroCash has also been deployed as zcash[5] but the system required an initial setup of almost 1GB data of proving key and verifying key for zero-knowledge proof which is costly but achievable as shown in [9].

Among the cryptocurrencies, Ripple takes a relatively different approach from Bitcoin. Ripple [62] provides the functions of an RTGS, and at the same time, currency exchange of a traditional PTN. Fig. 3 shows the components of the network where the gateways A and B, usually financial institutions such as banks, introduces liquidity into the systems. They create issuances that represent the external tokens to transfer in the network. The gateway A issues USD.A and EUR.A while the gateway B issues USD.B and EUR.B. Out-of-band the other nodes of the network deposit some "real world" money at the gateways so that they can declare a trust-line with a gateway to accept (a maximum value of) their digital currency and uses the currency for transactions with other parties.

An interesting direction stems from a recent Bank of England report [7] which covers several salient questions for the financial industry given the developments in technology. One of the primary concerns that central banks may have is loosing control of money-supply as an instrument of policy. This specific question is tackled directly by RSCoin [24]. This network delegates the monetary supply to a central authority, such as a bank, but utilizes a distributed network of other parties to perform transaction validation. The RSCoin network claims to provide strong transparency and audit capacity combining a central monetary provider with the attractive features present in a

[5]https://z.cash.



Gateways A and B corresponds to financial institutions and provide liquidity, i.e. create issuances that represent the external tokens to transfer in the network. The client X, Y, U, and V declares their trust-line to the gateways A and B. The actual deposit of external values at the various gateways is external to the system as in the e-cash protocol. For instance user X accepts at most 1000 USD.A.

Figure 3: The Ripple Network Components and Gateways with the External World

distributed transaction system (chiefly decentralized ownership). However, it suffers from the fact that it relies on the Central Bank to merge the high level blocks and that would create a central point of failure let alone a computational bottleneck . RSCoin can be basically considered as a traditional digital PTN with cryptography extension borrowing from Bitcoin. The Central Bank is still on the top of the hierarchy and maintains the ultimate ledger.

4. High level features of Payment Transactions Networks

Simple inspection of the developments shows that new proposals tend to be capability driven aiming at improving some features of an existing frameworks without challenging the overall design or purpose. To challenge the "feeling of being necessary" behind each architecture, we decompose the foundational requirements of a transaction system and then categorize the current engineering approaches against those requirements.

High Level Actors. A typical transaction process must involve at least the main actors: a *Payer* (who pays) and a *Payee* (whom to be paid). In a centralized PTN, a *Central Authority* (CA), who is *trusted* by Payer and Payee, is needed to consolidate transactions into the PTN and, upon realization, receive some fees. In a decentralized PTN, clearing and settlement might be performed by *untrusted Brokers* (e.g. miners in Bitcoin).

For the avoidance of doubt, in this scenario we do not consider the "physical" identity of the entity behind the nominal identity in the PTN.

Example 4.1. Taken on its own, a Bitcoin address has no less (and no more) anonymity than a traditional bank account number. It is only the information held by the bank in its enterprise information system, usually *outside* the payment network, that allows to associate a 27 alphanumeric international bank account number to a physical or legal entity.

High Level Steps. Currency, first of all, needs to be created (by CA or Brokers) before circulation. The value creation of currency is often taken for granted[6] despite having a *complex and long history* [37]. Currency then can be transferred from Payer to Payee. To describe the further steps of a payment process, it is useful to start from the definition usually accepted in the business domain:

> "After a sender *submits a payment* message to a payment system, the message must pass through that system's validation procedures. Validation will vary by system and can include security measures, such as verification of the sender's identity and the integrity of the message. If a payment is determined by the system to be valid, the system then typically checks whether necessary conditions for settlement are satisfied, such as the availability of sufficient funds or credit for settlement. Payments that pass the conditionality test are prepared for *settlement.* Under some payment system frameworks, settlement finality (that is, when settlement is unconditional and irrevocable) occurs when the receiver's account is credited." [55] (See section 2.1.1 p.5).

From this definition, it is clear that a payment process needs two conceptual steps of first "making a promise for a payment" by the Payer submitting a message into the system and then the "fulfillment of promise" by the CA or Brokers for the settlement finality. Certainly, the currency value are never consumed but always preserved to make future payment possible hence the last necessary step is the "preservation of value".

Therefore, we identify four critical steps of a digital PTN in common with traditional ones are: creation, promise, fulfillment, and storage.

4.1. The 4 steps of payment: Creation, Promise, Fulfillment, and Preservation

In a PTN, a conceptual step is centralized if we need the CA to perform a particular action (e.g. the central bank signs the coin-base transactions and broadcasts it to the miners in RSCoin). Without the CA's actions, the step is not valid, no matter what the others do (e.g. the miners cannot generate a coin-base transaction without the CA's signature in RSCoin).

On the contrary, a step is decentralized if more than one actor is needed to perform some actions for the step to be valid. If they don't act the step is not valid, no matter what the others do (e.g. in RSCoin, the CA cannot simply "add" a transaction into the ultimate blockchain, the transaction has to be the result aggregated from the miners; on the other hand, the miners' transactions have to arrive into the ultimate blockchain to be considered as concluded).

Table 2 summarizes actors interactions in these conceptual steps.

Example 4.2. E-cash is centralized while Bitcoin and ZeroCoin are completely decentralized. RSCoin and Ripple are hybrid systems where the Creation is centralized and Brokers need to *agree with CA unconditionally* on the deposited value of a Payer/Payee but Brokers are present to perform clearing and settlement (in Promise, Fulfillment and Preservation). In the Promise step, CA is usually not required and only involved in the Fulfillment step.

[6]For instance fiat currency is always issued by a CA, i.e. the central bank.

Table 2: Actors Involvement

In Creation and Preservation, the role of Payer and Payee are used interchangeably since they behave the same. We only distinguish Payer and Payee in a transaction, i.e. Promise and Fulfillment.

	Centralized	**Decentralized**
Creation	*Payer* makes out-of-band deposit and *CA* credits the account balance or issue tokens to Payer. If *Brokers* are present (Ripple, RSCoin), they will also be *notified by CA*.	*Payer* makes out-of-band deposit and Brokers collectively *agree* on the new account balance or tokens of Payer.
e.g.	E-cash, Ripple, RSCoin	Bitcoin, ZeroCoin
Promise	*Payer* gets payment address from *Payee* and sends payment information to *CA*.	Similar actions by *Payer*/*Payee* but payment information is sent to Brokers, CA is however *not necessarily notified* even if present (Ripple, RSCoin).
e.g.	E-cash	Bitcoin, ZeroCoin, Ripple, RSCoin
Fulfillment	*CA* validates the payment information then, upon successful, credits the *Payee* or issues new tokens to Payee.	Similar actions but performed by *Brokers*, if CA is present, *CA* will maintain the final result (RSCoin) unless there are multiple CAs (Ripple).
e.g.	E-cash	Bitcoin, ZeroCoin, Ripple, RSCoin
Preservation	*Payer* stores authentication secret, *CA* stores account balance or valid/spent tokens.	*Payer* stores authentication secret, *Brokers* (and *CA* if present) store account balance and valid/spent tokens.
e.g.	E-cash	Bitcoin, ZeroCoin, Ripple, RSCoin

***Creation* of value** This conceptual step involves putting value into the payment system for circulation. The two basic actions required are (1) "out-of-band" deposit where all parties agree on some value being created and (2) payment-token creation where value is represented via tokens, or account credit where the balance of a Payer is recognized. The payment-token is usually referred as currency.

Example 4.3. For E-cash, Ripple, and RSCoin, the value creation is done through a CA (a bank) as they exchange the fiat currency into digital cash, i.e. the CA decides the validity of the currency, while Bitcoin and ZeroCoin achieve the value creation through "mining" and out-of-band deposit the PoW into coins (in Bitcoin, upon finding a PoW, the miner is rewarded with 25 BTC [56], see Table 7), i.e. all (majority-honest) Brokers need to agree on the same set of coins being created. For hybrid systems such as Ripple and RSCoin, Brokers are only present in value creation to receive information from CA (e.g. the amount made available to a Payer for validation purpose in the next steps).

***Promise* of payment** refers to the action of announcing that a Payer is willing to transfer value to a Payee. As shown in Table 2, Payer needs to present the authentication secret while Payee is needed for the payment address in payment information. CA will be notified in centralized PTN whereas in decentralized systems only Brokers get notification from Payer/Payee.

Example 4.4. In terms of cheque payment, that is the action when the payer presenting the payee a cheque whereas in a traditional digital payment that is a payment request the payer sends to the central authority. In E-cash, the Payer sends the token to the Payee to present to the CA while the payment request is broadcasted by the Payer to all Brokers in Bitcoin, ZeroCoin, Ripple and RSCoin.

***Fulfillment* of transactions** For "physical" currencies, the exchange is an instantaneous fulfillment of the promise. However, in digital PTN, whenever a "promise" is made, it needs to be fulfilled. Firstly, the transaction's validity is ensured through validation rules and time-stamping. The validation rules cover the procedure to validate a transaction. It requires a check of the payer's identity, balance, and sometimes, the authenticity of the currency. The time-stamping of the transaction is critical to ensure a correct order of the transaction ledger. A centralized PTN requires only the CA to perform the fulfillment while a decentralized PTN requires the Brokers. A hybrid system requires both.

Example 4.5. E-cash only requires the actions of the CA while Bitcoin, ZeroCoin, and Ripple need only brokers (see Table 2). However, for RSCoin, the settlement finality (the ultimate ledger) is maintained by the CA.

***Preservation* of value** usually involves storing two kinds of data. The first one is the balance of a PTN client's account (for account-based PTN). The second kind of data is the transaction log which allows the audit of the transaction history. The transaction log provides the transparency in the currency network and prove that some transactions are fraudulent or not, e.g. the Bitcoin blockchain is also the transaction log storage. This is important for both token and account-based PTNs. To store value, Payer is always involved to store the authentication secret, as shown in Table. 2. Decentralized PTN requires the Brokers to store the transaction history whereas Centralized PTN requires a CA to store the account balance or the set of available/spent tokens.

Example 4.6. Bitcoin and ZeroCoin require the the brokers to store the coins and their corresponding public keys while the Payer stores the private keys. Differently, E-cash includes only a CA in the process to store the spent coins. For Ripple and RSCoin, both brokers and CA are involved. This step is carried out mainly by Brokers and CA where they have to verify the validity of the transactions and do record keeping.

For further details, we refer the reader to Appendix A where we present a comparative analysis of the steps for (online) E-cash, Bitcoin and its alternative, ZeroCoin, Ripple, and the centralized currency RSCoin.

5. Threats to Payment Transaction Networks

For each of the systems, besides *integrity* of value, we consider two additional important properties: *confidentiality* and *anonymity*. Confidential means that the owner is known but the value may not be known while anonymous means that the owner is unknown while the value is known.

Integrity. For this type of threat, the criteria for threats classification are whether the others benefit from the victim's loss and whether the victim is actively involved in one's loss. We identify three high level threats: loss (systemic and individual), fraud, and theft. See Table 3.

The threat of losses due to individual sloppiness or misfeasance are also present in traditional PTNs and can be broadly characterized in the following classes whereas the example countermeasures can be found in Table 4:

Over-Drafting When the payer promises a transaction whose value exceeds one's available funds.

Table 3: Loss of value

Notes: The first threat to a distributed PTN is the possibility of a party to lose money. A key distinction is whether such losses can be due to systemic failures or to individual actions either by a victim's own lack of care or by malicious activities of other parties. From the table it is clear that distributed PTNs make individual thefts harder as it requires changing the value of the global ledger, whilst systems which include a role of the payee in the transaction make losses due to lack of care easier. However, technological failures to achieve consensus (eg. Bitcoin forks) introduce systemic risk akin to the failure of a central bank.

Threats	**Others Benefit**	**Victim Actively Involved**	**Examples**
Systemic Loss	-	-	*CA* fails and all values are lost; Blockchain forks.
Individual Loss	-	yes	Lose the Serial#, private keys.
Fraud	yes	yes	Over-drafting; Over-spending.
Theft	yes	-	Unauthorized-spending.

Table 4: Loss of value countermeasures

To mitigate Over-Drafting, Double-Spending and Theft, validation rules are needed while Individual Loss is usually prevented by multiple (and partial) backups of the authentication secrets.

System	**Over-Drafting**	**Double-Spending**	**Individual Loss**	**Theft**
E-cash	*CA* check *Payer*'s balance before signing blinded Serial#	*CA* stores spent Serial# and check upon deposit.	*CA* stores the last n batches of blinded Serial#, send back to *Payer*. *Payer* unblinds the data.	N/A
Bitcoin	*Brokers* check that input $\geq$ output	*Brokers* check the inputs are unspent & blockchain is hard to rewrite.	N/A	*Brokers* check the private key.
ZeroCoin	Same as Bitcoin.	*Brokers* check the serial numbers are unspent & Blockchain is hard to rewrite.	N/A	*Brokers* check the Serial#.
Ripple	*Brokers* verify balance $\geq$ output	N/A	N/A	*Brokers* check the signature.
RSCoin	Same as Bitcoin.	Same as Bitcoin.	N/A	Same as Bitcoin.

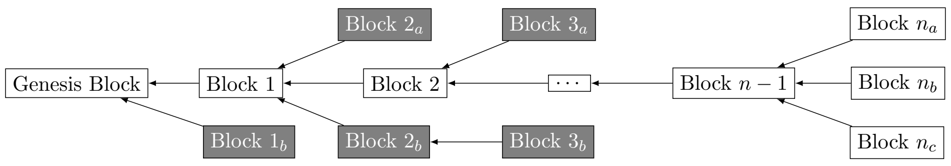
Double-Spending When the payer can use a token twice.

Unauthorized-Spending A payer spends another one's money or changes the value in another's account.

Individual Loss This can happen when the E-cash or ZeroCoin owner loses the serial numbers. Anther possibility is that the Bitcoin or RSCoin network users lose their private keys.

Systemic Loss A systemic loss in a PTN is not caused by an individual but rather by some intrinsic fragility of some components of the system itself.

In the presence of centralized CAs, the obvious sources of fragility is the trustworthiness of



Notes: After $n-1$-blocks, three concurrent transactions (n_a, n_b, n_c) have been proposed concurrently. The global chain forked and has to wait until some payer decide to append a transaction onto one of the three alternatives. Some previous forks in the past at block 2 resulted in orphaned transactions (in grey). Those are *not realized* in the system, no matter whether they were actually valid from the perspective of a "normal" payment system. The situation is particularly dire for those along the forked chain $3_b : 2_b : 1$: they have added transactions along what they thought was an eventually legal chain. The payers need to re-play some transactions along the entire subchain again (both 2_b and 3_b) if they want the transactions to be inserted in the final ledger.

Figure 4: The Systemic Risk of Blockchains: Forked and Orphaned Transactions

those very CAs. An example is the failure of Central Banks. By design, CAs can print money into the system, and therefore protection against their failure must be dealt with outside the protocol.

While all token-based payment systems suffer from systemic risk such as attacks on crypto, e.g. quantum computing, distributed PTNs are amenable to an additional different type of systemic risk, namely the *failure to reach consensus*[7]: the blockchain forks and some legitimate transactions are not included in the main chain. We illustrate this scenario in Figure 4 for the particular case of Bitcoin. At some point in time several nodes are generating new blocks or proposing transactions. The member of the PTN closer to the promising payers will create different blockchains. In that case, the blockchain is forked and every nodes will maintain the fork until conflict resolution mechanisms kicks in.

Example 5.1. In Bitcoin, the consensus mechanism is based on the presence of a "longer" chain. The longer chain becomes the main chain and the other tentative chains can be considered invalid. The blocks in the invalid chain are "orphaned" blocks. From the perspective of the distributed PTN, they are all potentially valid but payment promises. This happens irrespectively of the actual validity of the promises from the view point of authenticity of the payers requests and the availability of funds at the payers' accounts. This threat is *not* present in traditional PTN.

Every transactions in the orphaned blocks needs then to be put back to the transaction queue to be added in the new block. As in Fig. 4, the blocks 1_a, 2_a, 2_b, 3_a, and 3_b are orphaned transactions as a result of previous fork resolutions. At block $n-1$ the chain suffers from three forks (blocks n_a, n_b, and n_c) and has to wait for the next fork resolution. Another threat of systemic incoherence is when the outcome of a transaction is different from individual to individual in the network [10].

Confidentiality and Anonymity. The second type of threats, information disclosure, involves answering three questions.

Instantaneous Networth At time t, can an attacker identify the total value v of a nominal identity I?

[7]We do not consider the scenario of dishonest majority here since it implies the complete breakdown of the distributed system.

Table 5: Information disclosure countermeasures

System	Instantaneous Networth	Transient Value	Persistent Identity
E-cash	Only *CA* knows the balance.	Encrypt coin data with *CA*'s key.	*CA* keeps the balance DB private.
Bitcoin	NO	NO	*Payee* uses different key pair per transaction.
ZeroCoin	*Payee* applies coin minting.	NO	*Payee* uses different key pair per transaction.
Ripple	NO	NO	NO
RSCoin	NO	NO	Same as Bitcoin.

Table 6: Potential and Realized Threats

System	Realized Threats	Potential Threats
E-cash	Lose both Serial# + blinding factor.	N/A
Bitcoin	Loss of coins (invalid PubKey or losing PriKey). Selfish-mining. Exchanger fraud.	Majority attack.
ZeroCoin	Same as Bitcoin.	Same as Bitcoin. Loss of Serial#
Ripple	Disagreement	Majority attack. Gateway fraud.
RSCoin	Loss of coins (invalid PubKey or losing PriKey). Deanonymization. CA & Exchanger fraud.	Same as Bitcoin. The final blockchain (only one copy at the Central Bank) is rewritten by CA.

Transient Value At time t, can an attacker know about a transaction of total value v between two nominal identities I_1 and I_2?

Persistent Identity Can an attacker link two nominal identities I_1 at time t and I_2 at time $t+1$?

The example countermeasures against loss of either confidentiality or anonymity are summarized in Table 5.

Potential Threats. Beside the threats which we have just mentioned, there are still potential threats related to the technology and some of them have been actually realized. These threats need to be handled outside the payment system.

Selfish-mining Selfish-mining is an attack on the block mining mechanism that allows unfair block distribution (hence unfair *value creation*) . [30].

CA or Exchanger Failure Exchangers are third-party organizations that provide exchange for the digital currency and another currency such as commodity, fiat or another digital currency [12]. When the CA or Exchanger fails, the deposited funds are simply lost.

We summarize those threats in Table 6. Other threats also exist. The instability of a currency's value, inflation, or the realization of another digital currency can drive the participants away from a currency network. Some national governments may try to control or even ban the digital currency from usage in the country because they have no control over it, and the anonymity of the digital currency allows illegal activities (e.g. money laundering, illegal financing, and evading taxes).

Table 7: Required Cryptographic Actions

There is no required crytographic actions for the Preservation phase.

	Centralized	**Decentralized**
Creation	Payer *generates* a token. CA *verifies* and *signs* the new token. Payer *verifies* the new token.	Payer *generates* the new token; *Brokers verify* the new token.
Crypto Tech.	digital signature, blind signature	commitments, Merkle tree
Promise	Payer *generates* and *encrypts* the payment information (authentication secret, transfered amount, Payee's address).	Payee *generates* the payment address. Payer *generates* the payment information (authentication secret, transfered amount, Payee's address).
Crypto Tech.	public-key crypto	public-key crypto
Fulfillment	CA *validates* the payment information then *generates* new tokens for Payee. Payee *verifies* the new tokens.	Brokers *verify* the new token of Payee.
Crypto Tech.	blind signature	public-key crypto, (cross-)hash, Merkle tree, zk-proof
Preservation	-	-

6. Technological Components

We now briefly summarize the technological components of digital PTNs. First we describe the cryptographic primitives for proving authenticity and integrity as well as maintaining user anonymity, following by the consensus protocols providing decentralized agreement for the digital ledger.

6.1. Cryptographic Primitives

Table 7 summarizes the required cryptographic actions in the 4 conceptual steps of payment and make examples of the possible cryptographic primitives for each step[8].

Digital signature [40, p. 439] is used to provide some form of authenticity to payers and payees.

Example 6.1. Bitcoin, ZeroCoin, and RSCoin (token-based) user receives the coins via an address in the form of the hash of a public key. Later, to unlock the funds, a user has to provide a signature generated with a corresponding private key to prove the ownership of the coins. Ripple (account-based), however, use digital signature only to authenticate the user as s/he has to sign the transactions with the private key before sending the transactions to a validator. This also applies to the gateways in their transactions for value creation. Vanilla digital signatures are also used by RSCoin to authenticate messages from "bank-endorsed" participants (i.e. mintettes) in the consensus protocol.

In the E-cash scheme, the blind signature [19] allows the CA sign the serial number without knowing its content[9]. However, the E-cash model only offers the ability to create fixed-value coins. To allow the different value coins, the central authority can introduce various coin keys, each of which, upon signing, stating that the coin has a specific pre-defined value. A decentralized e-cash

[8]For the sake of description, we only provide a summary of the technology, we refer the reader to the technical paper for concrete details

[9]Technically this is done by multiplying a blinding factor. After the central authority has signed the blinded serial number with its private key, the content is unblinded by dividing the signed message with the blinding factor.

The transaction that creates the value is the coinbase transaction T_0 which send 10 BTC to the public keys address A_{pub}, 8 BTC to B_{pub} and 7 BTC to C_{pub}. In order to spend those coins, in transaction T_N, the owners have to prove they process the private keys A_{pri}, B_{pri} for the output index 0 and 1 of the transaction $T0$ respectively. This also applies to transaction T_M with D_{pub}, D_{pri} and E_{pub}, E_{pri}.

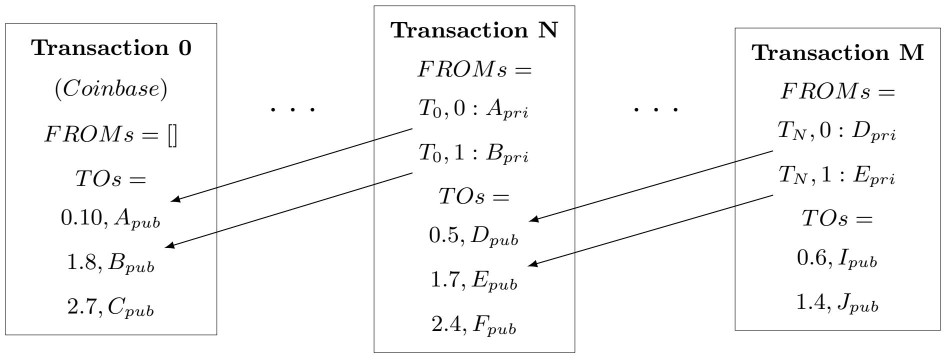


Figure 5: The *simplified* Bitcoin Transactions

scheme [54] is used by ZeroCoin to maintain anonymity yet still allow a user to prove ownership of a minted coin.

Hash functions [40, p. 153] are typically used to guarantee the integrity of the ledger. A blockchain is simply a sequence of applications of a hash function to a sequence of transactions. Every block contains the information of the current transactions and a reference to its previous block header's hash.

Example 6.2. Fig. 5 outlines the *simplified* process of conducting transactions in the Bitcoin network. There are 2 types of transaction: a coinbase and a regular transaction. While the latter one indicates some BTCs are being transfered from an address to another, the former is for the introduction of new coin into the system for circulation (value creation). As such, it is a coinbase transaction that distributes a miner's reward. Each block in the blockchain is included with a set of transactions. Each transaction consists of a vector of inputs and a vector of outputs. Each input of a transaction T spends a certain amount of BTCs from a previous output of a concluded transaction T_0. To spend the previous output, the input references a tuple of $(hash, n)$. A *double-SHA256* is performed on the T_0's raw data for the *hash*, while the n is the index of the output in T_0. Each output of a transaction T specifies an amount that will be transfered to a Bitcoin address. A block also hashes all of its transactions into a Merkle Tree [52] to maintain integrity.

Cross-hashing [61] is used by RSCoin which also uses a signed hash from the central bank to mark epochs in the chain. Due to lack of space, we refer to [40] for some further discussion of crypto systems.

Commitment scheme [35] allows a user to commit a secret v with some randomness r to create the commitment c. Later, to prove that c is a valid commitment of v, it suffices for the user to reveal v and r. Such scheme provides the properties of hiding and binding. Informally the hiding property prevents an attacker to learn the secret value v given only c. The binding property, on the other hand, makes it impossible for the user to open c in two different ways, i.e. the user cannot find $v' \neq v$ and $r' \neq r$ to form the same c.

Table 8: Communication and Consensus Requirements

There is no requirement for the Preservation phase.

	Centralized	**Decentralized**
Creation	Payer *sends* deposit to CA. CA *sends* tokens to Payer (in token-based PTNs) or *broadcasts* to Brokers if present. *CA makes decision* and all Payees (and Brokers if present) agree on the amount/tokens available to Payer.	Payer *broadcasts* deposit information (e.g. PoW) and Brokers validate and *broadcast* new amount/tokens of Payer for collective agreement.
Promise	*Payer and Payee* agree on payment info (authentication secret, transfered amount, Payee's address). Payee *sends* payment address to Payer. Payer *sends* payment information to CA.	Payee *sends* payment address to Payer. Payer *broadcasts* payment information to Brokers. *Payer, Payee and Brokers* agree on payment information (authentication secret, transfered amount, Payee's address).
Fulfillment	*CA makes decision* and all other actors agree on the amount/tokens available to Payee. CA *sends* new tokens to Payee (in token-based PTNs).	All *Brokers collectively agree* on the validity of the transaction and the new amount/tokens available to Payee. *CA* is also notified.
Preservation	-	-

Merkle tree [53] can be used in conjunction with zero-knowledge proofs for commitments to guarantee the anonymity of a user in a payment system. Throughout the execution of the system, a Merkle tree based on a collision-resistant hash function, where the leafs are commitments is maintained and updated. As in [64, 8], the number of leafs is not fixed a-priori, but it is possible to efficiently update a Merkle tree by appending a new leaf, resulting in a new Merkle tree; this can be done in time and space proportional to just the tree depth. The supported operations are adding a node to the tree, returning an authentication path from the root to a value, and a verification function that return true if the a path is authentic and false otherwise.

6.2. Consensus Protocols

Table 8 summarizes the communication and consensus requirements in the 4 conceptual steps of payment. If a step is decentralized, Brokers are present hence a broadcast channel is required besides the point-point channel. In addition, the Brokers need to run a distributed consensus protocol to collectively agree on a decision made on value creation, transaction and preservation.

The classic distributed consensus problem can be described as a set of n participants, out of which t are possibly faulty, seeking to agree on a single value v drawn from a set of value V proposed by the participants. The most trivial solution would be majority voting. The classic paper on Byzantine agreement on synchronous system [46] defines three properties that an agreement protocol must hold; (i) Validity: The single value v must be initially proposed by non-faulty participants, (ii) Agreement: Every non-faulty participants must agree on the same value v and (ii) Termination: The agreement must be reached in finite time. The robustness of a Byzantine protocol is measured in terms of the number of faulty participants that it can tolerate. It is proven that Byzantine agreement is only possible with $n \geq (3t + 1)$ [46].

A further FLP impossibility result, applicable to deterministic and asynchronous system [32] has proven that even with a single faulty-participant, the distributed consensus cannot be achieved in asynchronous settings because the nodes fail to differentiate a failure from a delay. To circumvent the impossibility results, researchers introduced techniques such as randomization[60], failure detectors [18], or assume partial synchrony [27]. In such systems, safety is always prioritized over liveness.

The Bitcoin network introduced a solution to a variance of the Byzantine agreement problem. The mechanism of adding blocks allows the happening of "forks" where disagreement arises within the network. The Bitcoin network's consensus problem is different from classic ones as the number of nodes in the network is exceptionally large and dynamic. The consequence is that nodes cannot decide on a value of majority. The nodes will vote for the "chain with most work" according to the blockchain by mining on the next block referencing to that chain's last block. Additionally, the block generation rate is technically maintained at 10 minutes to allow the propagation of block across the whole network so that every node shares the same blockchain. This solution practically allows the tolerance of faulty-nodes up to 50%. However, a consequence is that the transaction clearing speed is unacceptably slow for any financial implementation (e.g. the Chicago Mercantile Exchange marks to market millions of trades in less than a minutes) even though several variants of the protocol are introduced in [67] and [29] to improve the transaction clearing speed.

Ripple's agreement is achieved through the *Ripple Protocol Consensus Algorithm* [66] (RPCA). In each round, each server will collect the transactions, validate them, and then broadcast the valid transactions as candidates. The candidates will be voted for veracity by the other servers in the server's unique-node-list. If a transaction achieves at least 80% of agreement, it will be put in the ledger. The ledger will then close for current round and become the last-closed ledger for next round. RPCA can only tolerate up to $n \geq 5t + 1$. A comprehensive analysis on traditional and PoW-BFT can be found at [72].

Ripple, and similar systems, have not yet accounted for all possible cases that must be faced by real payment systems. One of the problem is churn when servers leave the network for maintenance or other reasons. In this case, the number of faulty nodes may exceed the threshold and compromise the ledger. If this is perceived as realistic threat by the network stakeholders then the incorporation of a stronger consensus algorithm [6] may be needed. In the cited work, the authors managed to tolerate up to j churning servers where $n \geq 5t + 3j$. Furthermore, some crypto PTNs are fully asynchronous (e.g. Bitcoin), whilst real systems actually have a pre-defined synchronization point, e.g. the Chicago Mercantile Exchange Globex Futures synchronize all trades between 13:59:00 and 14:00:00 Central Time for each trading day (see §(7)). For such systems, a synchronous protocol could be more suitable such as [73].

7. Beyond Simple Payment Networks: Derivatives-contracts-as-programs

Smart contract is not a new concept. As mentioned in §2.1, it has already proposed in [39] and implemented in traditional financial system [65]. For distributed systems, the notion of "smart" contracts was posited by Bitcoin and some other blockchain-based distributed ledgers e.g. Ethereum's Decentralized Autonomous Organization (DAO). The term "smart contract" is used to describe a type of contract that is *self-enforced* through the use of computer system, hence "smart" in the term. Technically a smart contract contains a set of execution instruction describing the functionalities of a programmed-value PTN, e.g. a fund-crowding campaign or a futures contract.

Example 7.1. The first smart-contract-supported DAO, "TheDAO" was launched as a venture capital funding in May 2016. The crowd-funding was $150 million at peak value. TheDAO is supported by and stored entirely in Ethereum currency units (ETH). The objective of TheDAO was to create a venture capitalist fund designed to initiate other projects and demonstrate the creation of DAOs, see `daohub.org`.

Indeed, the contract in this manner is simply stored as *passive* data in the blockchain. The contract must then be executed by the provider of the PTN infrastructure itself (e.g the miners of Ethereum). Hence, those smart contracts are essentially indistinguishable from the Eber's original proposal of smart contract [39]. The current prevailing smart contract implementation replicates a bond or dividend payment (and only in cryptocurrency, e.g. ETH in Ethereum). In this case the contract is pre-programmed with a regular payment date and recipient until the maturity of the debt. Hence, the only information the contract needs after its inception is the date and time.

A more complex approach that covers a wider variety of financial instruments requires some measurement of ex-post state of a contingent variable, such as a stock index, commodities price or reference interest rate. In this section we will discuss the natural extension to the standard distributed ledger which contains passive information, which we coin "derivatives-contracts-as-programs": that is active executables within the distributed framework. Hence, the ledger evolves to a distributed database with active execution and the programmed-value PTN may become non-monotonic as any of its component may have dynamic features. To accommodate the extension, we first discuss two additional features on top of the already mentioned high-level features (for simple PTN, in §4). We further make an example of a distributed futures exchange utilizing the new features. Finally we move on and elaborate the new possibilities and new threats of derivatives-contracts-as-programs in §7.2.

7.1. Distributed implementation of Derivatives-contracts-as-programs

A futures contract is a standardized agreements between two parties to buy or sell an underlying asset, at a price agreed upon today with the settlement occurring at some future date [68]. They are "promises" to buy or sell, and these "promises" can themselves be traded. Such trading is conducted in a double auction market operated by a centralized clearing house called Futures Exchange [2], such as the Chicago Mercantile Exchange (CME) [10]. Traders can 'quote' a future by specifying a price and notional volume of assets at which they will buy or sell (a limit order), or initiate a trade by placing a market order for a "promise" of a quantity (purchase or sale) at the best price from the standing quotes.

A futures exchange, e.g. the CME platform, can be implemented in a distributed PTN if we associate a future contract to an executable program. The current literature already associates a simple program to a derivative contract and use blockchain to guarantee the integrity of the program in the PTN.

As we have already noted, this is not sufficient to unleash the full potentiality of the system. For the association of *derivatives-contracts-as-programs* to work properly and reflect accurately the functions of the CME, we need two additional properties of a contract beside the code itself: parameters and parties bindings.

Parameter bindings The market prices of the contract corresponds to the input of the program and the distributed PTN must bind it to the market or source where such values needs to

[10]On the CME, futures contracts range from bushels of corn to Euro/US$ exchange rates. Recently, CBOE and CME launched Bitcoin futures markets. These are 'cash-futures', that is as they are settled in cash. Eurodollars futures are the largest world market by notional volume: in quadrillions of dollars/year. See https://en.wikipedia.org/wiki/List_of_futures_exchanges.

A simple contract

```
ContractID CLH0-345698-32456
input marketPrice;
party seller;
party buyer;

const tradeVolume = ...

buyer.cash = buyer.cash
- marketPrice * tradeVolume;
buyer.volume = buyer.volume + tradeVolume;
seller.cash = seller.cash
+ marketPrice * tradeVolume;
seller.volume = seller.volume - tradeVolume;
```

Its bindings

```
ContractID CLH0-345698-32456
marketPrice = .... crude oil in March 2020 ....

ContractID CLH0-345698-32456
buyer = jw@192.152.234

ContractID CLH0-345698-32456
seller = cnn@superbank.com
```

Notes: A simple derivatives-contract-as-program. The market price is bound to the crude oil in Mar 2020 market. The parties buyer/seller are bound to the trader's recognized ids. The initial step is essentially identical to the promise one for PTN and can use the same types of validation checks used by brokers to validate a transaction from Payer A to Payee B. Additional checks might be required by the distributed exchange (e.g. mark to margin). Brokers executing the contract to mark it to market must also ensure that the sum of the gain/loss of two outputs at the time of daily settlement amounts to zero. This latter check is implicit in the controls in the Fulfillment step, but must be made explicit here.

Figure 6: A simple contract and its parameters/parties binding

be extracted. The blockchain technology provides an "in-house" solution in this extent, i.e. by providing the record keeping of the quotes (both sell and buy), the spot prices of the contract are to be derived from the information on the blockchain. In practice, Ethereum smart contracts can alleviate the burden of computing the contingent variables through an *out-sourcing* platform called Oraclize[11] which provides access to data sources (defined as trusted providers of data). The data source can be as simple as a website, e.g. Reuters, the CME data stream, etc. or as complicated as verifiable computations running in virtual machine or Trusted-Execution-Environment. However, comparing to the secure in-house solution, the our-sourcing approach still needs rigorous security assessment.

Parties binding to traders' positions The output of the program must then be bound to traders accounts in the PTN. From an operational perspective, this is simply a list of pairs (trader's nominal identity in the PTN and a corresponding output element of the program). Once again blockchain can be used to guarantee the integrity of the binding.

Example 7.2. With this set up we can now transfer the CME functionalities to a distributed PTN (see [**?**] for a concrete implementation). Sell and buy positions corresponds to the value of the corresponding output variables as illustrated in Figure 6.

For *settlements*, each broker of the PTN executes the code of the contract as embedded in the blockchain, by reading the input values from the bound reference (i.e. deriving the spot prices from the quotes recorded on the blockchain or consulting the data source) and computing the corresponding output values. As the contract includes the binding of the parties to the nominal

[11] http://www.oraclize.com/.

identities of the traders in the PTN, each broker then uses the consensus algorithm of the PTN to initiate the corresponding transactions (either crediting or debiting the results). After the transactions are finally realized the accounts of the traders are aligned to the market value as they would be in a centralized exchange.

The *trading of a derivative contract* in the CME simply corresponds to change in the bindings between parties of the contracts and the corresponding nominal identities in the PTN. This simply requires the owner of a (sell or buy) position in the contract to initiate a transaction as a Payer of the blockchain and the corresponding Payee is the the new owner of the position. Each broker would register the "transaction", i.e. the change in ownership of the derivatives contract's positions, after having verified its validity (e.g. by marking to margin). A part of the steps are already detailed in §4.1's Fulfillment but additional steps specifically aimed at guarantee the balance of the ledger must be implemented.

7.2. New possibilities and new threats

New possibilities and new threats also come with the introduction of dynamicity in derivatives-contracts-as-programs.

Dynamic programs for PTN. Table 9 summarizes the new possibilities in both static and dynamic programs. A dynamic program may introduce dynamicity also in data structure and the Promise and Fulfillment process may be carried out multiple times before finally terminated.

In Creation, the data structures are *fixed* and the data is *initialized* in a static program, yet a dynamic program allows the reference to a *not-yet defined* virtual structure. Similarly, the parameters of a static program is a *constant* at time of Promise while a dynamic program binds the parameters to an *external source* which may depend on the outcome of the previous executions of the program *after the Promise phase.* The parties binding step of a static program determines the Payer at the time of Promise whereas the Payee can be set at time of Fulfillment. Differently, a dynamic program may change the parties binding at different times depending on the execution.

When a static program is chosen as a contract, a Promise step is also static. In this case, the contractual process of the chosen contract gets started and partially executed by some parties. The control is then transfered to the other parties in Fulfillment. Finally the process is terminated once and for all. On the contrary, when a dynamic program is bound to the contract, as it gets started in the Promise step, the contractual process may be a continuation of a partially executed process[12] and will only terminates after several executions (see King Of The Ether[13]). As a result, the Preservation of a static program only requires the storage of data defined at Creation and modified by Promise and Fulfillment while a dynamic program requires additional storage of *contractual processes status.*

New threats. In the exchange example above, an obvious issue for checking the validity of a trade is whether the Payer is actually the entity mentioned in the binding. These checks are already discussed in §4.1's Fulfillment step. Additional check may generate distributed exchanges of different nature. They could mimic the checks performed by the centralized exchange mechanism.

[12] A layman example will be an employment contract promising to pay for several years in a row a net amount to the worker, as opposed to a gross amount. The taxes and social security charges will be functions dynamically provided by the government.

[13] https://www.kingoftheether.com/thrones/kingoftheether/index.html.

Table 9: Dynamicity of programmed-value PTN

	Static	**Dynamic**
Creation	Data structures *determined*; data *initialized*	May contain reference to some *not-yet-defined* virtual data structures
e.g.	A futures trader enters the market and her margin is set to a specific initial value	The margin is a pointer to a yet to be initialized digital wallet
Parameters binding	Evaluated to a *constant* value at time of Promise	Bound to an *external source* and may be evaluated after the Promise phase
Parties binding	The Payer is *determined at Promise* and the Payee is *determined by Fulfillment*	The parties *can be changed at different times* depending on the execution
e.g.	A classical transaction where Payer specifies the Payee and the transfer amount	A trader in futures market posts a quote to buy some futures contracts from a (*not-yet-known*) seller at market price (only known at time of Fulfillment)
Promise	A *static* program is chosen as a contract and a contractual process is *started* and partially executed by some parties	The chosen program is *dynamic* and the contractual process may be the *continuation* of a partially executed process
e.g.	A standardised futures contract where the Payer has to pay an amount equal to the market price times the volume of contracts	A known standard futures contract where the price is determined by invoking a function to specified by a third party
Fulfillment	The execution of the process is transferred to all the other parties and finally *terminates*	The (partial) execution is carried-out by the other parties and only *swapped-out* after execution. This will *repeat several times* before finally terminates
e.g.	The Payee adds funds into her balance then terminates	A futures trader has to restart the net position computation every time the order book is updated by a quote, this will only terminates at final settlement
Preservation	Only data defined at Creation and modified through Promise and Fulfillment	Have additional contractual processes status
e.g.	The balance and the authentication secret of the Payer/Payee in a classic payment network	A non-standardised futures contract in which execution needs to be resumed by the Payer after Fufillment to meet additional obligations unless value be lost, e.g. options or the King of Ether being reset after 14 days

For examples, if traders are not allowed to bid for contracts whose current payout exceeds their margin, each broker in the PTN would have to execute the contract and make sure that the Payee has enough capacity to stand the losses with the current value (as opposed to the value at time of settlement, or the previous year average, etc.). A whole variety of different arrangements is possible provided they only required to execute the contract from the bound inputs and possibly check the resulting outputs against the current account of the new owner.

New threats also become possible: *malicious (or buggy) contracts* that makes it possible to create a Ponzi scheme or even a cluster of schemes that mutually feed each other. Consider as an example the contract in Figure 6 where the instruction `buyer.cash = buyer.cash - ...;` is deleted. Clearly the execution of that contract would generate a money pump for the holder of the Buy position . To avoid such cases, an additional requirements must be checked: brokers executing the contract to mark it to market must ensure that the *aggregated* outputs amount to zero. This check is essentially implicit in the current implementation of distributed PTNs, but it must be included as a explicit protocol step in the contract-as-program PTN. An infamous example was the Ethereum DAO mishaps [22], several other attacks are introduced in [5] and [47] regarding buggy implementation of contracts. An attack on futures market DAO is also described in [50].

Additionally from a security perspective a dynamic derivative-contract-as-program is clearly an instance of a *multi-party reactive security functionality* [16]: every agent must satisfy individual constraints (usually monotonic) and the system as a whole must satisfy global constraint (possibly non-monotonic). The global constraint may be such that an agent's legit move can unpredictably make another (honest) agent's state invalid due to the change in the public information and the global state, e.g. a trader in the futures market makes promise to buy some contracts at daily settlement but could not meet it due to market price changes (by other traders' quotes).

This means that *new type of failures* are possible and the protocol might need to consider the option of *honest failures and failures by omission.* In fact failures by omission is important in financial intermediation, especially for non-monotonic protocols [51].

Example 7.3. In a distributed futures market, as no one but Alice herself can *prove the validity* of her trading inventory, whenever a new order arrives and changes the market price she has to publish some (cryptographic) proofs to mark her inventory valid [49]. If she discovers that her inventory is *not valid* (hence her *honest failure*) and learns that she *cannot benefit from participating* in the protocol anymore, she would simply stop joining the step and the protocol is halted waiting for her messages. Thus a protocol to handle *failures by Alice's omission* is desirable.

We will examine this facet of active contracts in follow up work.

8. Conclusions

In this paper we have provided a systematic review of both traditional payment systems and their analogues in decentralized ledgers. We have dissected the different technological features and associated them to the corresponding business and financial requirements. In this way we have highlighted the essential characteristics of a distributed Payment Transactions Networks and, above all, identified possible novel combination of these features.

In particular, a key property of a distributed Payment Transactions Network is to ensure that the ledger can guarantee global consistency against nodes that either by malice or by sloppiness undermine the correct functioning of the ledger (for instance to double-spend). Failure to address such issue at the forefront of any distributed payment system is a road to a systemic failure akin to the failure of a central bank in a traditional system.

An interesting follow-up of this observation is the idea of distributed computations, whereby the transactions in the blockchain need not be simply stored but actually be executed. We discussed the idea of a *derivatives-contract-as-program* that is marked to market (or an account that is margined) automatically by computations run on, and whose ownership transitions are recorded, in the distributed ledger. This proposal opens up new possibilities and new challenges.

References

[1] Dennis Abrazhevich. Classification and characteristics of electronic payment systems. *Electronic Commerce and Web Technologies*, pages 81–90, 2001.

[2] Franklin Allen and Anthony M Santomero. The theory of financial intermediation. *Journal of Banking and Finance*, 21(11-12):1461 – 1485, 1997.

[3] Milton Anderson. The electronic check architecture. Technical report, Financial Services Technology Consortium, 1998.

[4] Apple. Apple pay. `http://www.apple.com/apple-pay/`, 2015. Accessed: 2015-12-30.

[5] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts. Technical report, Cryptology ePrint Archive: Report 2016/1007, https://eprint. iacr. org/2016/1007, 2016.

[6] Roberto Baldoni, Silvia Bonomi, and Amir Soltani Nezhad. *An algorithm for implementing BFT registers in distributed systems with bounded churn*, volume 6976 LNCS of *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 32–46. 2011.

[7] Bank of England. One bank research agenda. `http://www.bankofengland.co.uk/research/Documents/onebank/discussion.pdf`, 2015. Accessed: 2015-12-30.

[8] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[9] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 287–304. IEEE, 2015.

[10] Bitcoin.org. 11/12 march 2013 chain fork information. `https://bitcoin.org/en/alert/2013-03-11-chain-fork`, 2015. Accessed: 2015-12-30.

[11] Bitcoin Wiki . Bitcoin mining introduction. `https://en.bitcoin.it/wiki/Mining#Introduction`, 2015. Accessed: 2015-12-30.

[12] Bitcoin Wiki . Bitcoin mt. gox. `https://en.bitcoin.it/wiki/Mt._Gox`, 2015. Accessed: 2015-12-30.

[13] Bitcoin Wiki . Bitcoin proof of stake. `https://en.bitcoin.it/wiki/Proof_of_Stake`, 2015. Accessed: 2015-12-30.

[14] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 104–121, May 2015.

[15] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology–EUROCRYPT 2005*, pages 302–321. Springer, 2005.

[16] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multiparty secure computation. In *Proc. of the ACM STOC*, pages 494–503. ACM, 2002.

[17] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[18] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267, 1996.

[19] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1982.

[20] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[21] CME Group. Eurodollar. `http://www.cmegroup.com/confluence/display/EPICSANDBOX/Eurodollar`, 2015. Accessed: 2015-12-30.

[22] CoinDesk. Understanding the dao attack. `http://www.coindesk.com/understanding-dao-hack-journalists/`, 2016. Accessed: 2016-06-25.

[23] Wei Dai. b-money. `http://www.weidai.com/bmoney.txt`, 1998. Accessed: 2015-12-30.

[24] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. Technical report, University College London, 2015.

[25] John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.

[26] Evan Duffield and Daniel Diaz. Dash: A privacy centric cryptocurrency. 2014.

[27] Cynthia Dwork, Nancy Ann Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[28] Ethereum. A next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2015. Accessed: 2015-12-30.

[29] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol.

[30] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[31] Hal Finney. Rpow - reusable proofs of work. `https://cryptome.org/rpow.htm`, 2004. Accessed: 2015-12-30.

[32] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[33] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *Public Key Cryptography–PKC 2007*, pages 181–200. Springer, 2007.

[34] Seth Gilbert and Nancy Ann Lynch. Perspectives on the cap theorem. *Computer*, 45:30–36, 2012.

[35] Oded Goldreich. *Foundations of Cryptography – Volume 2: Basic Applications*. Cambridge University Press, 2004.

[36] Google. Google wallet. `https://www.google.com/wallet/`, 2015. Accessed: 2015-12-30.

[37] Friedrich August Hayek. *Denationalisation of money: The argument refined*. Ludwig von Mises Institute, 1990.

[38] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols (extended abstract). In *Secure Information Networks*, volume 23 of *IFIP — The International Federation for Information Processing*, pages 258–272. Springer US, 1999.

[39] Simon Peyton Jones, Jean-Marc Eber, and Julian Seward. Composing contracts: an adventure in financial engineering. In *ACM SIGPLAN Notices*, volume 35–9, pages 280–292. ACM, 2000.

[40] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

[41] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 2013.

[42] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 839–858. IEEE, 2016.

[43] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41. ACM, 2014.

[44] Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to use bitcoin to play decentralized poker. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 195–206. ACM, 2015.

[45] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 406–417. ACM, 2016.

[46] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[47] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 254–269. ACM, 2016.

[48] Nancy Ann Lynch. A hundred impossibility proofs for distributed computing. In *Proceedings of the eighth annual ACM Symposium on Principles of Distributed Computing*, pages 1–28. ACM, 1989.

[49] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams. Futuresmex: Secure, distributed futures market exchange. In *2018 2018 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 453–471.

[50] Fabio Massacci, Chan Nam Ngo, Jing Nie, Daniele Venturi, and Julian Williams. The seconomics (security-economics) vulnerabilities of decentralized autonomous organizations. In *Proceedings of the twenty-fifth Security Protocols Workshop*, 2017.

[51] Fabio Massacci, Chan Nam Ngo, Daniele Venturi, and Julian Williams. Non-monotonic security protocols and failures in financial intermediation. In *Proceedings of the twenty-fifth Security Protocols Workshop*, 2017.

[52] Henri Massias, Xavier Serret-Avila, and Jean-Jacques Quisquater. Design of a secure timestamping service with minimal trust requirement. In *the 20th Symposium on Information Theory in the Benelux*. Citeseer, 1999.

[53] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987.

[54] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013.

[55] David C Mills, Kathy Wang, Brendan Malone, Anjana Ravi, Jeffrey C Marquardt, Anton I Badev, Timothy Brezinski, Linda Fahy, Kimberley Liao, Vanessa Kargenian, et al. Distributed ledger technology in payments, clearing, and settlement. 2016.

[56] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Unknown, 2008.

[57] Nxt.org. Decentralized financial ecosystem. `http://nxt.org`, 2013. Accessed: 2015-12-30.

[58] PayPal. Paypal. `https://www.paypal.com/us/webapps/mpp/home`, 2015. Accessed: 2015-12-30.

[59] Colin Percival. Stronger key derivation via sequential memory-hard functions. Self-published, 2009.

[60] Michael O Rabin. Randomized byzantine generals. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 403–409. IEEE, 1983.

[61] Michel Raynal and Mukesh Singhal. Logical time: Capturing causality in distributed systems. *Computer*,

29(2):49–56, 1996.

[62] Ripple Labs. Executive summary for financial institutions. `https://ripple.com/solutions/executive-summary-for-financial-institutions/`, 2015. Accessed: 2015-12-30.

[63] Nicolas van Saberhagen. Cryptonote v 1.0. `https://cryptonote.org/whitepaper_v1.pdf`, 2012. Accessed: 2015-12-30.

[64] Tomas Sander and Amnon Ta-Shma. Auditable, anonymous electronic cash. In *Annual International Cryptology Conference*, pages 555–572. Springer, 1999.

[65] Lexifi Sas. Structuring, pricing and processing: Complex financial products with mlfi. *White Paper, Aug*, 2003.

[66] David Schwartz, Noah Youngs, and Arthur Britto. The ripple protocol consensus algorithm. Technical report, Ripple Labs Inc White Paper, 2014.

[67] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoins transaction processing.

[68] Daniel F. Spulber. Market microstructure and intermediation. 10(3):135–152, 1996.

[69] Nick Szabo. Bit gold. `http://unenumerated.blogspot.it/2005/12/bit-gold.html`, 2005. Accessed: 2015-12-30.

[70] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123.

[71] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. 2014.

[72] Marko Vukolić. *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*, pages 112–125. Springer International Publishing, Cham, 2016.

[73] Xianbing Wang, Yong Meng Teo, and Jiannong Cao. Message and time efficient consensus protocols for synchronous distributed systems. *Journal of Parallel and Distributed Computing*, 68(5):641 – 654, 2008.

Table A.10: Creation of Value

Creation of value involves putting value into the payment system for circulation. The two steps required are (1) out-of-band deposit and (2) payment-token creation. The payment-token is usually referred as currency. As we can see, both E-cash, Ripple, and RSCoin requires the ability of all nodes in the network to recognize the signatures of CAs and, possibly more importantly, acknowledge the authority of CAs to introduce value into the PTN. The only difference between those systems is how many CAs can actually participate.

System	**Central Authorities**	**Brokers**	**Payer**	**Payee**
E-cash	Sign the Serial#. Balance account. Send coin to *Payer*.	N/A	Out-of-band deposit value into *CA*. Blind and send the Serial#. Unblind the CA-signed coin.	-
Bitcoin	N/A	Verify PoW, add the block. Broadcast the block.	Solve the PoW and then broadcast a coin-base transaction.	-
ZeroCoin	N/A	Verify PoW and ZeroCoin. Add and broadcast the valid block.	Solve the PoW and then broadcast a coin-base transaction. Mint Bitcoin to Zero-Coin.	-
Ripple	Sign transactions for creation of issuances. Broadcast a transaction to send value to the *Payer*.	-	Declare a trust-line upon the *CA*' tokens. Out-of-band deposit value into *CA*.	-
RSCoin	Sign transactions for creation of issuances. Broadcast a transaction to send value to the *Payer*.	-	Out-of-band deposit value into *CA*.	-

Appendix A. Comparative analysis of example PTNs

In this section we present a comparative analysis of the steps for (online) E-cash, Bitcoin and its alternative, ZeroCoin, Ripple, and the centralized currency RSCoin. These systems are selected as they are illustrative for features and fairly easier to comprehend comparing to their variants. Their features are summarized in a dedicated Table for each step of a payment system[14].

Appendix A.1. The Creation of Value

Table A.10 compares the key steps of the 5 systems subject of our analysis.

In E-cash, the client withdraws coins from the bank, out of the client's account, by sending a blinded serial number to let the bank signs the corresponding blind signature to the information.

In the Bitcoin network, upon solving a Proof-of-Work, the solver is rewarded with some bitcoin (BTC). The BTC is bound to the owner by an address, i.e a public key whose private key is only known to the owner. The Proof-of-Work puzzle must be hard to solve but easy to check. It must also be a progress-free puzzle, which means the chance of solving doesn't increase with effort. This process is referred to as "mining" and the individual nodes are commonly referred to as "miners"

[14]For simplicity in description, we only mention the mechanism and refer the reader to the respective papers for implementation details.

[11]. Once they successfully solve a block puzzle, they will receive a reward and all transaction fees included in the block. For the Bitcoin system, this reward is 50 bitcoins (BTC) in the first instance. Then it will be halved every 210.000 blocks. Currently, by the time of writing, the reward is 25BTC. Ultimately, until the year of 2140, the reward will be reduced to 10^{-8}BTC.

ZeroCoin [54] is based on Bitcoin. The protocol provides additional anonymity for the network users by allowing the user to "mint" the basecoin (BTC) into a ZeroCoin. To achieve this, the user generates a random serial number s and a random number r, then s is encrypted into the ZeroCoin z with r. The minting operation requires a Bitcoin transaction that spends d BTCs, which is also the value of the ZeroCoin z, plus a small amount of transaction fee. The authors argue that ZeroCoin provides real anonymity compare to the pseudonyms in Bitcoin due to the fact that the pseudonyms' transactions are traceable in the blockchain.

In Ripple, the gateways, usually banks, create issuances to transfer in the network but the payer must first deposit value into the gateways. To ensure the integrity of the issuances, the gateway will sign the issuances with their private keys. The clients declare a trust-line with a gateway to "trust" and accept (a maximum number of) its digital currency and uses the currency for transaction.

RSCoin [24] simply delegates the value creation to a Central Bank. The Central Bank is a Central Authority and assumed to be honest. To generate value, the Central Bank signs a transaction to allocate the value to an address.

Appendix A.2. The Promise of Payment

Table A.11 illustrates how payment is initiated.

The E-cash scheme's payment is initiated as the payer send the coins to the payee. The coin's data is encrypted with the bank's public key so that the payee cannot peek the coin's serial number.

In the Bitcoin network, a payment is started with the payer creating a transaction. The payer specifies the unspent transaction outputs to use as inputs in a new transaction. The outputs are the address of payees with the respective amount of coin. The transaction will be broadcast into the whole Bitcoin network to notify all nodes.

ZeroCoin requires an additional step to initiate the payment. For a payer to spend the ZeroCoin, the payer must first construct a transaction to claim an unspent mint transaction output as input. The output of the transaction is the same as Bitcoin's. The unspent mint transaction output is claimed by providing the serial number s and the random number r as a zero-knowledge proof.

Ripple's users also create a promise of payment via transactions. A Ripple transaction must be signed by a user's private key. The transaction must include the address of a payee and the specific amount of tokens along with their issuer ID. Then the transaction is broadcast into the whole Ripple network for validation.

RSCoin's payment promise is similar to Bitcoin's. In the RSCoin network, there exists a large number of nodes, called mintettes, authorized by the central bank via PKI-infrastructure. The payer must create a transasction and send it to the corresponding mintettes to prove that the payment input is unspent and thus for them to include the transaction into a block.

Appendix A.3. The Fulfillment of Transaction

Table A.12 shows how payments are validated and the PTN converges to a new state where the value of the transactions is finally accrued to the payee and subtracted to the paye's account in the PTN.

The E-cash scheme requires the original minting bank to perform the verification of the coins. When a payee receive the encrypted coin data from the payer, the payee forwards the data to the

Table A.11: Promise of Payment

The *Promise* of a payment is the action by a payer to announcing to the PTN a transfer of value to a payee. In terms of cheque payment, that is the action when the payer presenting the payee a cheque whereas in a traditional digital payment that is a payment request the payer sends to the clearing house. It is a promise as eventual fulfillment is not initially warranted. The payee plays a relatively minor role and the requirement of some protocols to use a private key to pocket the transaction could be eliminated altogether, when the transactions are public. There would be no need to "unlock" the fund, as the whole PTN could simply acknowledge that the payee is the recipient. Only when the payee needs to use the received funds he would need a private key. Yet, when this would happen s/he will be a payer. . .

System	**Central Authorities**	**Brokers**	**Payer**	**Payee**
E-cash	-	N/A	Encrypt the Serial#, send the data to *Payee*.	Get data from *Payer*
Bitcoin	N/A	Receive the transaction	Create a transaction with a value and *Payee*'s address, then broadcast the transaction. *Payer*'s private key is needed to unlock the available funds.	Provide the public key as payment address.
ZeroCoin	N/A	Receive the transaction	Same as Bitcoin but requires the *Payer* to apply spending function on the stored Serial# to unlock the available funds.	Same as Bitcoin
Ripple	-	Receive the transaction from *Payer*.	Sign a transaction and broadcast. The transaction includes address of *Payee*, the amount, and issuer of the tokens.	Same as Bitcoin
RSCoin	-	Same as Bitcoin	Same as Bitcoin	Same as Bitcoin

Table A.12: Fulfillment of Transaction

During the fulfillment step, the promise to pay is finally accrued to the payee and subtracted to the payer's account in the system. It requires checking the payer's balance and possibly the authenticity of the currency. A time-stamping of the transaction may be needed to ensure a correct order of the transaction ledger. A critical issue in distributed PTNs is that every broker must be able to check the balance of the payer and broadcast the result to achieve a consensus on the ledger. Suitable incentives must then be in place to guarantee the timely cooperation of a sufficient number of brokers.

System	**Central Authorities**	**Brokers**	**Payer**	**Payee**
E-cash	Check the spent Serial# database for double-spending. Credit the *Payee*. Mark the spent Serial#.	N/A	-	Forward data from *Payer* to *CA*.
Bitcoin	N/A	Validate the transaction by checking the public and private key of the available funds. Verify that transaction inputs are previous unspent outputs. Solve the PoW, add the transaction into a block, then broadcast the block.	-	-
ZeroCoin	N/A	Validate the transaction by applying the verifying function Check that the Serial# is not in spent database. Solve the PoW, add the transaction into a block, then broadcast the block.	-	Mint Bitcoin to ZeroCoin.
Ripple	-	Validate the transaction. Broadcast the validation result and add the transaction into ledger upon agreement .	-	-
RSCoin	Aggregate the result from the *Brokers*' low-level blocks to form high-level blocks and add into blockchain.	Validate the transaction by checking the public and private key of the available funds. Then send the block to *CA*.	-	-

minting bank. The bank will decrypt the coin data with its private key. The serial number will be checked within the spent serial number database to prevent double-spending. If everything is correct, the bank will accept the coin data and credit the payee's account with the corresponding amount. The deposited coins' serial number will be added into the spent serial number database.

For the Bitcoin network, the miners receive the transactions from the payer and validate the correctness of the transactions before adding them into blocks. After solving the PoW and creating a block that includes the transactions, the miners will broadcast the block into the whole network. Other nodes, upon receiving it, will perform the validation for the PoW, the transactions' correctness, and the integrity of the block before adding it into their blockchain and broadcast their acceptance.

In the case of ZeroCoin, a transaction is realized after the nodes apply the verifying function to the ZeroCoin z and check that the serial number s is unspent. After the transaction is accepted, the payee will need to initiate a new minting transaction to create a new ZeroCoin from the transaction's output.

Table A.13: Preservation of Value

Notes: The *Preservation* of value usually involves storing two kinds of data. The first one is the balance of a PTN client's account. Whilst commodity and token currencies can be stored in physical form, a digital PTN must store value as numeraire in some ledger whose authenticity is recognized by all parties in the PTN. The second kind of data is the transaction log which allows the audit of the transaction history. Several systems are more vulnerable to theft or failures as they require to store data to unlock the funds as payees (as opposed to use them as payers). Further, distributed PTNs require all brokers to have access and to store the transaction log of the *entire* system.

System	**Central Authorities**	**Brokers**	**Payer**	**Payee**
E-cash	Store available funds of a *Payer* and Spent-Serial#.	N/A	Store the Serial#.	-
Bitcoin	N/A	Store the blockchain.	Store the private keys to unlock the unspent transaction outputs.	-
ZeroCoin	N/A	Store the blockchain. The global accumulator stores the binding of value and Serial#.	Store the Serial#.	-
Ripple	Store the out-of-band deposit of clients.	Store the consensus ledger Store the available funds of the corresponding *Payer*.	Store the private key to sign the transaction.	-
RSCoin	Store the "ultimate" blockchain.	Store the blockchain.	Store the private keys to unlock the unspent transaction outputs .	-

In a different approach, Ripple's consensus protocol requires more than 80% of nodes' agreement to consider a transaction valid and add it into the consensus ledger. On the Ripple network the nodes have to check for over-spending only. The payment is conducted by moving a certain amount of issuances from one account to another.

RSCoin requires the collaboration of mintettes and the central bank for the fulfillment of a transaction. Mintettes are authorized by the central bank to collect transactions and put them into the low-level blocks. The low-level blocks are periodically sent to the central bank to merge into a high-level blocks and add to the main blockchain. A transaction initiated by the payer, once sent to the corresponding mintettes will be validated before being added into low-level blocks. Since the mintettes are semi-trusted, PoW is no longer required to create a low-level block.

Appendix A.4. The Preservation of Value

Table A.13 summarizes the approaches to the storage of value in each selected PTN.

For E-cash, the balance of each payer must be stored in the minting bank's database to prevent over-drafting. To counter double-spending, a serial number database must be maintained to check the serial number of each deposit request. The payer must store the serial number carefully to use in a payment. The storage is only partially private as the minting bank knows the balance of each payer. Only anonymity of the payer in a transaction is guaranteed because the bank cannot trace back the original owner of the coin from the serial number.

Bitcoin miners store the whole blockchain in their hard-disk. The blockchain is everything in the Bitcoin network as every executed transaction is included in it. It is the same for the ZeroCoin network, except that ZeroCoin also needs a global accumulator which is a state-keeper for the binding between value and the secret serial number. The Bitcoin network users store private keys to unlock funds while the ZeroCoin users only store the serial numbers as they get rid of the private keys after finishing minting a ZeroCoin. The RSCoin network maintains low-level blocks at the mintettes and high-level blocks which form the final blockchain at the central bank while the users need to store the private keys to unlock funds from previous transactions. These three networks all provides anonymous but not private storage as every transaction is recorded in the blockchain. The anonymity of the storage is only provided if the payer use different pairs of public/private key for each transaction.

The Ripple network nodes store the consensus ledger for the balance of each payer and also the transaction history. Thus the payer only needs to store a signing key for the transaction as the Ripple network provides neither private nor anonymous storage.