Beyond Bitcoin – Part II: Blockchain-based systems without mining

Pasquale Forte[1], Diego Romano[2], and Giovanni Schmid*[2]

[1]PA Advice, Naples, Italy
[2]Cnr ICAR, Naples, Italy

July 28, 2016

"If the Lottery is an intensification of chance, a periodic infusion of chaos into the cosmos, then is it not appropriate that chance intervene in every aspect of the drawing, not just one?"
– Jose Luis Borges, *The Lottery in Babylon*

Abstract

Nowadays the decentralized transaction ledger functionality implemented through the blockchain technology is at the highest international interest because of the prospects both on opportunities and risks. There are a number of advantages inherently embedded in blockchain-based systems, and a pletora of new applications and services relying on concepts and technologies inspired by those of Bitcoin are emerging. But at the same time some weaknesses and limitations are evident, and we argue that they stem mainly from the fact that the blockchain is managed through mining. In this work we pointed out the unsustainability of mining in case of massive large-scale blockchain-based system. Moreover, after isolating the basic concepts behind mining, we sketched possible alternatives for the maintenance of blockchain-based systems. We considered security issues, incentives, as well as competitive and collaborative opportunities, and we proposed a framework of concepts and algorithms to implement blockchain-based systems for different contexts.

1 Introduction

In a previous work [1] we gave an overview of concepts and technologies grounding what we called *blockchain-based systems*. These are emerging systems intended to wide by far the scope of cryptocurrencies, and that use the ideas

*Corresponding author: Giovanni Schmid, email: giovanni.schmid@cnr.it introduced in the seminal work [2] for their core technologies. In particular, all these systems – although with minor variations in protocols and their implementations – are rooted in the concept of *mining*. The authority appointed to control transactions among parties in the network, which is the keystone of conventional systems, has been superseded in blockchain-based systems by a subset of nodes equipped with special software, called *miners*. Their job consists indeed in performing mining, a special kind of computational work which is mandatory for the recording of new transaction blocks on the blockchain.

At the current state-of-the-art, mining appears to be a strict requirement for any blockchain-based system. First and foremost, it is designed to protect the blockchain from tampering and, in the special case of cryptocurrencies, it allows for the creation of new coins in a controlled manner[1].
Another main feature of the consensus mechanism implemented through mining is the incentive for peers in managing the blockchain. Indeed, the miner (or a *pool* thereof) that is accepted by its peers as the winner of the computational race incorporated in mining is usually rewarded with some assets and (at the time being) with optional *transaction fees*.
Lastly, a less evident but important consequence of a mining-based consensus mechanism is the prevention of the following fraudulent threat: since, at least in the intentions of Bitcoin designer(s), it is very difficult to predict the winner of the mining race, then it is likewise difficult to blacklist them (e.g. through a denial-of-service attack) in order to steal the prize or to degrade the system performance.

By abstracting from its implementations, mining consists in a special workflow in which a miner node can give corroborate evidence to its peers that it worked on a computational problem - which is *difficult* to solve in some sense - and that its effort involved the processing of the last block in the blockchain alongside with a set of new transactions to be stored in the current block. The difficulty of the computational problem depends on a suitable *pricing function* [3] whose values are constrained by a *difficulty threshold*. The pricing function allows to instantiate a problem which can be made more or less difficult to solve but which cannot be made easier by previous calculations, even those in the current instance of the problem. Lastly, the difficulty threshold allows to tune the hardness of the problem so that a solution is moderately hard to find by the totality of miners [1] in a statistical sense.

Let us now recap the two currently main implementations of mining : proof-of-work and proof-of-stake systems.
In the context of blockchain-based systems, a *proof-of-work* [4] is a non-interactive proof protocol in which a prover demonstrates to a (set of) verifier(s) that she has solved a difficult computational problem, so that she can mine a new block. This problem is usually a kind of artificial puzzle of no practical concern – besides, of course, the existence of the network – which requires an increas-

[1]It could be useful to remark here that, as pointed out in [1], neither the blockchain architecture nor mining are strictly necessary in case the system provides just for coin (and/or resource) transfers without the creation of new coins. However, blockchain-based systems seem to be more efficient and scalable than systems based only on conventional digital signatures.

ing amount of computational resources over time, and where miners fitness is proportional to their hashing rate.
The *proof-of-stake* [5] mining approach, as an alternative, enables the stakeholders of the system to mine new blocks. In a proof-of-stake mechanism the difficulty of a mining activity is indeed proportional to the number of coins owned by the miner at current time: the hashing is timed in one digest calculation per second, and it must be calculated only once for each unspent coin in the wallet. It is worth to notice that a proof-of-stake cryptocurrency system requires a pre-distribution of coins at start-up, for which a proof-of-work protocol represents an "on-the-shelf solution". However, "pure" proof-of-stake systems are subject to the "rich gets richer" issue, and attempts have been made to combine the two approaches in order to mitigate risks related to one or the other (see e.g. [6]).

Although the development of recent blockchain-based cryptocurrencies aimed to mitigate the Bitcoin "Nonce Rush" [1] escalation experienced in these last few years, all the current blockchain-based systems rely on mining.

The main aim of this work is twofold: by one side we argue on the sustainability of mining in case of massive large-scale blockchain-based system, and on the other side we intend to show that the functional and inverse requirements for such systems can be achieved by means of alternative, cheaper workflows than mining. In such respect, we are going also to sketch a such possible alternative.

2 The economics of Bitcoin

Bitcoin introduced the blockchain concept and its related technology. It represents the archetype of the blockchain-based system and it is the most widely used one. Even if it arises in the cryptocurrency context, it can highlight some more general trends and for this reason we will shortly discuss it. At the present time, after several years of its existence, we have enough information and measures to start doing some evaluation of the impact of this system on the global economy, as well as sketching some future perspectives.

In [7] an extensive analysis of the sustainability of the Bitcoin network produced interesting results that we accepted in their magnitude, but in this context we propose some normalization to better understand some key points. Firstly let us consider the energetic impact of the Bitcoin network when compared to that of the common fiat currencies. In the following table we show a comparison between global Bitcoin and fiat money supply in terms of energy consumption (in GJ) and estimated value [2]. Looking at those figures, we can notice that Bitcoin

[2]In order to obtain these figures, we have performed the following simplified estimate. Let $\#BTC$ be the number of bitcoins in circulation and $ValBTC$ the price in $US\$$ of one bitcoin. We estimate the current money supply value of Bitcoin with the product of the two previous figures: $suppBTC = \#BTC * ValBTC$. At the moment of writing:

$$suppBTC = 15133150 * 393.43\$ = 5953835204\$.$$

Let $suppFiat$ be the estimated value of the global fiat money supply considering the current exchange rates. Then: $suppFiat = M0 + M1$, where $M0$ is the most liquid form of money

	Total Value ($)	Syst. Mgm. Cost (GJ)	Cash Mgm. Cost (GJ)	Total Cost (GJ)
Fiat	28.6×10^{12}	2318900000	39600000	2358500000
Bitcoin	6×10^{9}	3970000	0	3970000
% BTC/Fiat	0.00021	0.171201863	0	0.168327327

energy consumption measures only the 0.168327327% of the global fiat money consumption. But, on the other hand, Bitcoin represents only the 0.00021% of the global fiat money supply value.

This imposes us to make some projection and to imagine how could Bitcoin expand its energy consumption, in the fortunate case it would ever become the main global currency. Therefore, if we outline a scenario in which Bitcoin reaches the same value of the current global fiat money supply, supposing that the energy consumption of global Bitcoin would grow linearly with its value, the energetic impact of Bitcoin network at that point could be:

$$3970000 \times \frac{28.6 \times 10^12}{6 \times 10^9} \approx 19 \times 10^9 GJ,$$

that is the 802.4% of the energy consumption of the current global fiat money system. If this should ever happen, it would be absurdly inefficient.

Fortunately, to be fair, things will not move linearly in the next future, as we just grossly supposed. For example we should consider some progress in the design of computing hardware. Following Moore's Law, the number of transistors on a circuit-board will double every 18 months, but soon this will come to an end due to size constraints of silicon atoms. This implies a possible paradigm-shifting in computing architectures, with no reliable clues of its impact on Bitcoin mining.

At the same time, the Koomey's law [8] states that the energy needed for a fixed computing load halves every 18 months. This could be an encouraging signal that transactions validations in Bitcoin, which should happen every 10 minutes for the entire global community, could require less energy than today. But, to further complicate the possible scenario, in the last months the global Bitcoin hash-rate has experienced an exponential growth [9], that means that more and more computing power has been devoted to mining and transaction validation for speculative reasons.

Several economics professionals have a vision of what could happen to Bitcoin in the next future. According to the work of J.A. Kroll et al. [10], collaborative teams of Bitcoin miners, also known as *pools*, will give space to the expansion of powerful players. This could happen also because the increasing specialization in the hardware, with powerful application-specific integrated circuits (Asics), represents a barrier to enter the mining competition. Indeed, as

(coins and banknotes) and $M1$ represents Demand Deposits. On the other hand, at the moment of writing

$$M0 = 5 \times 10^{12}\$ \text{ (5 trillion\$)}$$
$$M1 = 23.6 \times 10^{12}\$ \text{ (23.6 trillion\$)},$$

thus $suppFiat = 28.6 \times 10^12\$$ (28.6 trillion$).

reported in CoinDesk State of Bitcoin and Blockchain 2016 [11], small pools tend to disappear over time, giving space to big players.

Even if the Henderson Rule of Three and Four [12] seems to not apply to IT and electronics, the current hashrate distribution [9] shows that two or three entities have more than 50% of the entire Bitcoin hashing power. Considering that Bitcoin is a system based on majority consensus, a cartel of miners could change any rules to uphold their mining strategy, or can even censor certain transactions. We believe that, at its current state, Bitcoin has serious risks that undermine its peculiar decentralization with possible scenarios which include a progressive desertion of the system.

We can conclude that the current mining vision is based on uncertain and risky forecast of energy management. Therefore the consequent trend in the aggregation of mining resources could lead to the success of individual speculative entities which control the entire blockchain.

3 Features and drawbacks of mining

Since our interest falls on generic blockchain-based system, before entirely turning away from the Bitcoin implementation and elaborating possible alternatives, we need to look over the mining in its role of tool for managing the blockchain. In particular, it is important to characterize the type of mechanism that is best suited to protect a blockchain, highlighting mandatory or desirable properties.

As we previously introduced, the mechanisms implemented so far consist in posing a challenge that comes in the form of a computational searching problem. It has the following features [13, 14, 2]:

- **Hard to solve** – The problem has to be hard to solve, otherwise too many peers could be eligible for creating new blocks. That in turn would result in too many branches during the updating process, and in a loss of consensus on a unique blockchain.

- **Easy to verify (system performance)** – A candidate solution has to be verified rapidly and in a cheap way to maintain a good system performance. However, the time required to verify the trustiness of a transaction - which in mining-based systems is proportional to the number of *confirmations* [15] received - seems to be important at the same extent or more.

- **Tunable difficulty** – The hardness of the problem must be dynamically adjusted by the system so that the overall set of miners can solve each instance in about the same time. This way miners are implicitly in sync in their competitive effort for the construction of the blockchain.

- **Unpredictability of the solver** – The more *mining power* a miner applies, the better are its chances to mine the new current block. However, at least in the absence of a *majority* of colluding parties, it has to be difficult to predict which would be the next solver of the problem. This means that no party (or pool thereof) can easily get control of the system, and that also miners with small hashpower fractions are incentivized in managing the blockchain. Moreover, in this way the miners that are eligible for updating the blockchain cannot easily blacklisted.

However, as it can now be desumed after the massive volume of research done on this topic in the last five years, systems based on the mining approach are prone to risks that are very difficult to mitigate in a truly satisfactory way. This is because some of the requirements listed above are somewhat conflicting (e.g. hardness of the challenge vs. system performance or vs. unpredictability of the solver), and they are unlikely to be achieved and maintained in an open environment where governance is realized through consensus, and where peers can behave selfishly also by colluding in a variety of ways.

For example, in [16] it is shown that, for a mining pool using the current implementation of Bitcoin and having a good control on information flow, the revenues earned using a special mining strategy called *selfish-mine* grow superlinearly with the pool size when compared to the revenues using an honest strategy, no matter what is the initial size of the pool. This way, a selfish pool could eventually grow to become a majority and control the system, even if the initial fraction of its hashing power is arbitrarily small. To prevent the success of such strategy, the authors of [16] propose a modification of the Bitcoin protocol setting up a threshold of 25%, that is a value by far under the well known theoretical threshold of 50%, showing that the Bitcoin protocol - despite such modification - cannot be safe against the infringement of the so called "Unpredictability of the solver".

Let us also remark that the "Tunable difficulty" requirement entails a problematic trade-off between challenge hardness and system effectiveness. As the challenges become harder, the peers are less incentivized in mining, and the mean time required to verify a transaction through an adequate number of confirmations[3] becomes longer, affecting system performance. Conversely, if the challenges become easier, the system performs better, but the consensus during the blockchain construction becomes more difficult to maintain when branching happens.

We can therefore state that the set of requirements for an effective mining is difficult (if not impossible) to implement and maintain in practice. Thus, two main questions are:

- what is the correct set of core requirements for blockchain-based systems?
- what are the most appropriate cryptographic mechanisms to implement the above requirements?

[3]This last number depends indeed only on the relative hashpowers of miners, not on the time required to find a nonce, as shown in [2].

4 An alternative to mining

In our opinion, an ideal blockchain-based system should reward its peers in proportion to their engagement in guaranteeing the integrity of the service (*due care*), not for their piles of resources (e.g. coins) owned within the system and/or any computational effort in solving a nonsense problem. Since checking for invalid transactions and assembling a valid transaction block (of appropriate size) are easy computational tasks that can be performed by *any* peer in the system, those cannot be used directly to assign a reward in the Bitcoin mining style. However, they can be used as a mandatory requisite for a peer in order to *get a chance* in having its block added to the blockchain (and eventually a reward for that). And the idea of "getting a chance in winning something" is tied to that of "raffle", not to "mining".
In this respect, we believe that the best principle is that of *equity*, which also results the most appropriate to meet the requirement of "Unpredictability of the solver". Indeed, it sounds correct to assign the same success probability to all the peers that are able to give corroborate evidence of their due care in building the new current block; on the other hand, the uniform probability distribution maximizes the uncertainty about the peer that will actually be in charge of adding the new current block.
This way the function of protecting the integrity of the blockchain is not tied to mining anymore, and proof-of-work or other computational tasks can be decoupled by the management of the blockchain, or even eliminated.

However, an ideal setting would have blockchain-based systems that rewards their peers for their effort in solving some useful and challenging problems. Actually, we have different systems of this kind to date [1], but no one seems to smartly conjugate the problem solving framework with a cryptocurrency. A smooth, effective integration would give rise a new impetus to collaborative computing, with great improvements in hardware/software resource utilization and power efficiency.
In the context of the "raffle approach" sketched above for blockchain protection, the incentives for peers to solve a given computational problem translate in a natural way into a number of raffle tickets granted to a peer in proportion to its effort in working at the problem. This way a new "proof-of-work" comes into play, but with a different meaning and purpose than in [4] and Bitcoin: indeed it proves that a peer actually worked on the right problem spending a precise amount of effort.

With this ideas in mind we believe that a better design for a blockchain-based protocol is a modular one consisting in what follows:

1. a *multiparty raffle protocol*, used to establish which is the peer A_j (from now preferably referred as *agent* instead of miner) in charge of adding the new block to the blockchain among the agents A_i selected by the following module (2), or module (3), or both;

2. a *proof-of-transcription scheme* used to give a publicly verifiable, corroborate evidence that a specified agent A_i assembled a new valid transaction

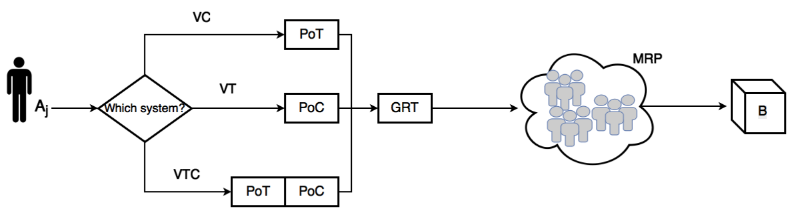

Figure 1: Overview of blockchain-based systems without mining.

block;

3. a *proof-of-commitment scheme* used to give a publicly verifiable, corroborate evidence that that a specified agent A_i worked on a problem that the system (through a coordinator/employer) wishes to support, spending on it a given amount of computational effort;

4. a *granting mechanism* which allows to express the amount of effort ascertained by modules (2) and (3) in terms of tickets for the raffle implemented by module (1).

This way the design can encompass a large variety of blockchain-based systems belonging to the following three main categories:

- systems that allow for a cooperative management of transactions related to one or more resources, and that can optionally provide for the creation of such resources. These systems require publicly verifiable corroborate evidence that the transactions put in each block are valid and that they are assembled according to prescribed rules. These systems are based on a proof-of-transcription scheme and therefore we will refer to them in the sequel as *Verifiable Transcription blockchain-based systems*, or *VT systems* for short;

- systems that allow for a cooperative resolution of one or more computational problems.The requirement for these system is to get publicly verifiable corroborate evidence about the amount of computational work performed by agents in relation to such problems. Since these systems require a proof-of-commitment scheme, we will call them *Verifiable Commitment blockchain-based systems* or *VC systems* for short [4].

- systems that allow for both a cooperative solution of one or more computational problems and the management of transactions related to one or more

[4] In this context the utility of blockchain technology stems from the following circumstances. There are usage scenarios, particularly when the solution of a problem is managed in a peer-to-peer way without any support by a trusted third party, where the resolution transcripts produced by peers must be validated by the network. In these cases, blocks can be used to encode publicly verifiable resolution transcripts, and the chaining of blocks is useful to keep track and to protect those transcripts from tampering.

resources. These systems require the coupling of a proof-of-commitment scheme with a proof-of-transcription scheme and they will be denoted as *Verifiable Transcription and Commitment blockchain-based systems*, or *VTC systems* for short.

Figure 1 sketches a possible high-level design for a system that would be able to cope with both the management of resources and the resolution of a given computational problem through a network of peers, but in a way that either cases can be optionally left out.
At this point, of course, it remains to establish which cryptographic primitives are the most appropriate to realize the previous modules, and how these primitives must be composed in order to get a secure protocol. The aim of the next section is to sketch a possible solution.

5 A possible realization

In the following we shall use both terms and notations specifically introduced for our proposal, and a bunch of symbols and terms inherited by the Bitcoin system, although for the last mentioned we could consider slightly different notions or implementations. Special symbols and terms that are specific of our design are summarized in Table 1, whilst for the remaining items and their meaning the reader will refer to [1], and in particular to the table of special symbols therein.

An *agent* A is a peer which can take part in building the blockchain and which can play the roles of *transactor* or *recipient* as well. Moreover, there can be special kinds of agents, depending on the specific implementation of the system (see Sections 5.2 and 5.3).

Some of the special terms and notions already present in Bitcoin – used here with a somewhat different meaning – are those of transaction, transaction block, block header, digest and threshold.
A *transaction* has the same structure than in Bitcoin, although it has now a mandatory field `issue`, whose value γ is set according to a local counter or timer of the transactor, and is protected through the transactor's signature.
A *transaction block* or simply *block* B has the same structure and function than in Bitcoin, but it lacks the *generation transaction*.
A *block header* H has a similar function than in Bitcoin, in that it serves both for realizing the chaining among the different blocks and for storing data which give corroborate evidence of the work performed by agents. However, in our design block headers encompass also the function of the generation transaction. Thus, by issuing H_j, agent A_j can redeem per se the transaction fees collected for its current block B_j, besides a predetermined grant in case the system provides for the generation of new assets. More generally, H_j allows A_j to indicate an amount of resources, benefits or coins as its potential reward, by means of:

- the transaction fees and a special H_j field accounting for a certain amount of new assets in a VT system;

- a special H_j field which accounts for a predetermined reward by the coordinator/employer in a VC system;
- the sum of the two previous incomes in a VTC system.

It could be useful to remark in this respect that block headers are data records whose fields are valued depending on the kind of blockchain-based system they have to support. In particular, block headers have a field for storing specific digests for proof-of-transcript tasks in the case of VT systems, and for proof-of-commitment tasks in the case of VC systems. On the other hand, VTC systems use two fields in order to keep track of both types of digests. We are going to detail the specific structure of block headers in the next sections, with respect to the kind of proof they are intended to.
A *digest* in our design is meant to be a fixed-length string obtained in a bunch of ways depending on the usage context, whereas in Bitcoin digests are solely obtained through cryptographic hash functions, i.e. functions having both the properties of *collision resistance* and *one-wayness* [17, 18]. Indeed, as we are going to detail later, digests D^T for proof-of-transcription schemes require only the collision resistance property, whilst collision resistance could be coupled or not with the one-wayness property for digests D^C in proof-of-commitment schemes, depending on the usage scenario (see Sections 5.1 and 5.2).
Finally, the *threshold* τ is a system-wide positive parameter as the difficulty threshold in proof-of-work related systems. However, it will be used in a different context and with a different purpose as explained in Section 5.3.

Two central notions in our approach are those of *witness* and *ticket*.

In the jargon of cryptographic protocols, a *witness* is a string sent by a party to another party in order to: (a) give corroborate evidence to a receiver that a given, univocally identifiable sender has done some private computation and/or has got some secret and (b) allow the sender to keep such private computation and/or secret undisclosed to the receiver until it has been verified [17, 19]. In our design, according to the above goals, witnesses W are (preferably fixed-length) strings returned by a function with the properties of one-wayness and *existential unforgeability* [20]. Such kind of cryptographic data can be efficiently computed from a wide range of inputs thanks to modern *signature schemes* [17].
Witnesses are computed by agents using a system-wide available signature scheme ($\texttt{sgn}, \texttt{vrf}$). Each agent is supposed to have a secret signing key uniquely bound to its public address A_j[5] on the system. Moreover, A_j is supposed to be able to privately compute the witness W_j by signing with algorithm `sgn` and its secret key (a message consisting of some of the fields of) its current block header H_j. Each other agent A_i is supposed to have got the public key uniquely bounded to A_j which allows to verify A_j signatures thanks to algorithm `vrf`. This way, when H_j will be disclosed by A_j, A_i can check if there is a mismatch between (the signed fields of) H_j and W_j. Witnesses allow for the chaining of block headers (and related blocks) through the inclusion in H_j of the field

[5]We use A_j to indicate both an agent and its address, when there is no risk of confusion.

Symbol	Meaning	Symbol	Meaning
A	Agent and its address	ν	Number of transactions in a block
$A^\star$	Super agent	γ	Current time
C	Commitment (amount of work)	δ	Delay time
D^C	Commitment digest	ρ	Number of tickets
D^T	Transcription digest	Γ	Granting function
W	Witness	Ω	Raffle function
P	Problem to be solved	$(\mathtt{sgn}, \mathtt{vrf})$	Signature scheme
O	Oracle or prover	$\mathtt{prg}$	Pseudo-random number generator
$(X)_2$	Representation base 2 of X	$\mathtt{hsh}$	Hash algorithm

Table 1: Special symbols and terms specifically introduced for the blockchain-based system proposed in the present work.

W^p which references to the previous block header, i.e. to the last block header registered on the blockchain. Moreover, the witness mechanism coupled with the digest fields allow an agent to attest the work done in relation to the current block B^j.

The notion of *ticket* is tied to the raffle protocol and it is a way to measure the effort spent by an agent in doing a proof-of-transcript, or a proof-of-commitment or both in terms of chances (or grants) to be rewarded for such work. Tickets are first and foremost a way to establish which agents can take part to the next "raffle extraction", but they can also represent an implicit mechanism through which agents get synchronized in the blockchain construction, as explained below. The *number of tickets* ρ will be calculated from the measured computational effort via a suitable efficiently computable *granting function* Γ.

We are going to give further details about the above notions and the other ones listed in Table 1 in the following sections.

One crucial point in the design of a blockchain-based system is to define the way in which the peers involved in the blockchain construction can synchronize each other so to interleave their efforts and give rise to a unique blockchain. It should be clear that such synchronization can be achieved through an explicit "shared clock" mechanism or – alternatively – through an implicit "average-time in doing work" mechanism analogous to that introduced in Bitcoin, which is tied to the difficulty threshold of a proof-of-work [1]. Getting a trusted, network-wide shared clock was presumably rejected in Bitcoin to avoid the use of a trusted third party with its related scalability and single-point-of-failure issues. However, we have nowadays direct experience of massively scalable distributed systems whose nodes keep their local clocks in sync effortless. Our smartphones and PCs do that thanks to loosely-coupling protocols like NTP [21]. On the other hand, managing raffle extractions in a peer-to-peer network seems a little bit more problematic than interlacing proof-of-work instances. Thus – at least in the context of some usage scenarios for VC and VTC systems – we prefer to rely on the stronger assumption of a *trusted global time* (see Section 5.3). When the above requirement is difficult to achieve in practice, as when it is troublesome to rely on a trusted third party, we can appeal to the implicit synchronization mechanisms offered through the notion of ticket. Indeed, an agent has to satisfy the condition “Minimum number of owned tickets to take part to the next raffle” before proceeding. Therefore agents can get synchronized with respect to such trigger points or, in case of VTC systems, the boolean AND between the previous condition and the one asking an agent to correctly complete its proof-of-transcription task. This way we have a synchronization mechanism similar to that implemented in Bitcoin; however, it drives agents to work faster in doing transactions management and/or in solving a problem assigned to them by a coordinator/employer, rather than resulting in the so much “eco-unfriendly” Nonce Rush.

In any case, on the basis of the above arguments, we will suppose throughout the sequel that agents can keep effortless their local clocks in sync within a reasonable tolerance[6] and without any detrimental effect on the scalability of the system and its fault-tolerance.

5.1 Proof of Transcription

Transactions can be used to keep track of transfers of many kind of resources, as exemplified by the many blockchain-based system introduced so far [1]. Each new transaction must be verified by agents in order to ascertain three facts: the identity of its transactor, whether the assets to be transferred are actually owned by the transactor, and whether only the recipients satisfying the conditions required by the transactor are accounted for such resources. *Transcription* consists in work performed by an agent in order to assure that the transactions managed through the system are valid and they are recorded in the blockchain as prescribed. We will assume that the number ν of transactions in a block is a system-wide publicly available parameter which the system can adjust dynamically in order to adapt its workload to the amount of global transaction traffic. Therefore, ν can be fixed over time similarly to the difficulty threshold of a proof-of-work in Bitcoin.

In the following we will suppose that the transactions announced on the system's network over time constitute a *tamper-proof totally ordered set*, i.e. a set $\{T^1, \ldots, T^h\}$ for which neither its elements nor the order in which they appear in the set can be changed by a unauthorized party. Such property can be easily obtained in an efficient way on the basis of the following criterion: (a) the lexicographic order of the system identifiers (*addresses*) of the agents that issued such transactions and (b) the value of the `issue` field for any two or more transactions having the same agent as transactor. The tamper-proof property follows indeed from the fact that each transaction is cryptographically signed by its issuer, so that its fields (included the `issue` and the `issuer` fields) cannot be changed by the other parties.

This way each agent will be able to group the transactions heard on the system network in blocks, each block being composed of exactly ν ordered trans-

[6]The NTP protocol allows for an accuracy of 10 ms over the Internet and a much better accuracy over smaller-scale networks.

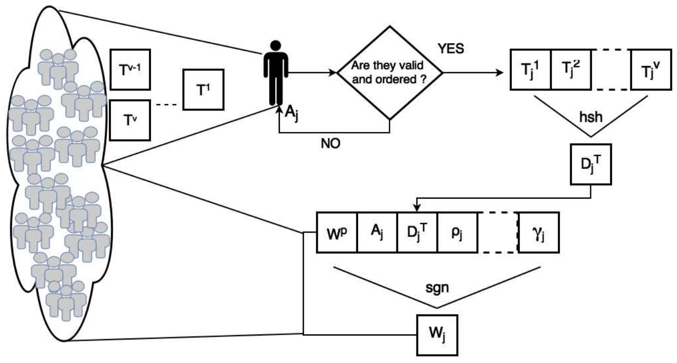

Figure 2: The proof of transcription workflow for agent A_j.

actions and no more data.
Let $\{T_j^1, \dots, T_j^\nu\}$ be the set of standard transactions collected by agent A_j by listening on the system's network and ordered on the basis of the above criterion. In order to proof its due care in building the current block B to be inserted in the blockchain, A_j proceeds as follows (see Figure 2).

1. A_j controls the validity of each element by checking its fields and signatures, and orders them correctly;

2. if the above verification step is passed then A_j computes a suitable digest for the set $\{T_j^1, \dots, T_j^\nu\}$, called *transcription digest* (denoted as D_j^T), which uniquely represents the above ordered set with overwhelming probability;

3. A_j composes its own current block header H_j which contains, among other fields, the witness W^p for the block header of the last valid block in the blockchain, its public address A_j, the current transcription digest D_j^T and the `tickets` and `issue` fields in the header instantiated with the values ρ_j and γ_j, respectively. Then A_j computes its current witness W_j by signing H_j;

4. A_j is now ready to take part to the raffle protocol.

Both fields `tickets` and `issue` have function and meaning that depend on the specific system. The ρ_j value is instantiated through the granting function Γ, whilst γ_j is the output returned by a local counter or clock.
As a consequence of the principle of equity described in Section 4, ρ_j will be 0 or a fixed value in the case of VT systems, since in this context it indeed represents a boolean flag indicating if agent A_j completed or not its transcription task with due care. A common and natural choice is $\rho_j = 1$ but, if the system has to provide for the creation of a certain amount $\rho > 0$ of new assets, that can be accounted through the assignment $\rho_j = \rho$.

The field `issue` is usually used to keep track of the instant of time γ_j in which header H_j is issued by agent A_j; however, this value is actually used for managing synchronization among agents just in some application scenarios where it is possible to rely on a source of trusted global time (see Section 5.3 for an example of this kind of usage).

Transcription digests D^T in this context play a role similar to the root digests $\widehat{D}$ in Bitcoin. Likewise, they can be obtained through *Merkle trees* [22] or some other kind of hashing algorithm `hsh`. However it can be useful to stress here that, according to the use of these digests in our design, D^T could be as well obtained by giving the ordered set $\{T_j^1, \ldots, T_j^\nu\}$ in input to a collision-free, order-preserving compression function (e.g. a lossless data compression algorithm).

5.2 Proof of Commitment

Commitment C consists in work performed by an agent in order to contribute to the solution of a computational problem performed by the system on behalf of a coordinator/employer (aka *super-agent*) $A^\star$. Let P be a computational or decisional problem which $A^\star$ must solve and for which $A^\star$ knows an *oracle* [23] or *proof system* [24] O so that each (partial) solution of P – or otherwise computational effort C in solving P – can be efficiently verified by evaluating $\mathsf{O}(C)$. Then $A^\star$ can ask the system (i.e. some or all the agents taking part in the network) a support in solving P, giving to the involved agents a chance to be rewarded proportionally to their effort. The commitment C is a suitable data which serves to measure the amount of job performed by each agent for solving P; e.g. it consists of the number of bit/bytes of a solution for P, a subset of its solutions, the set of values tested without success in a searching problem, and so on.

In any case we will assume that the super-agent $A^\star$ can set-up an efficiently computable *granting function* Γ such that, for any possible value assumed by C, $\Gamma(C) = \rho$ is a non negative integer. We will say that ρ is the *number of raffle tickets* since this integer establishes how many chances has an agent which performed the commitment C on P in being the winner of the raffle protocol (alternatively, ρ could indicate the grant given by $A^\star$ to the winner). In order to fix ideas and without loss of generality, we will suppose in the following that O is a verification system owned by $A^\star$ and which $A^\star$ can decide to release as public or to keep it private depending on its business, whilst the granting function Γ can be made available to all the agents without disclosing O. This way $A^\star$ can establish if it has to validate the commitments by itself (e.g. to keep them secret), or if they can be validated by agents A_i. Depending on these two cases, as detailed in what follows, $A^\star$ takes part or not in agent proofs-of-commitment and in the construction of the current block header.
In order to proof its current commitment C_j to the solution of P, A_j proceeds as follows (see Figure 3).

1. A_j performs a certain amount of work C_j in order to solve problem P, C_j

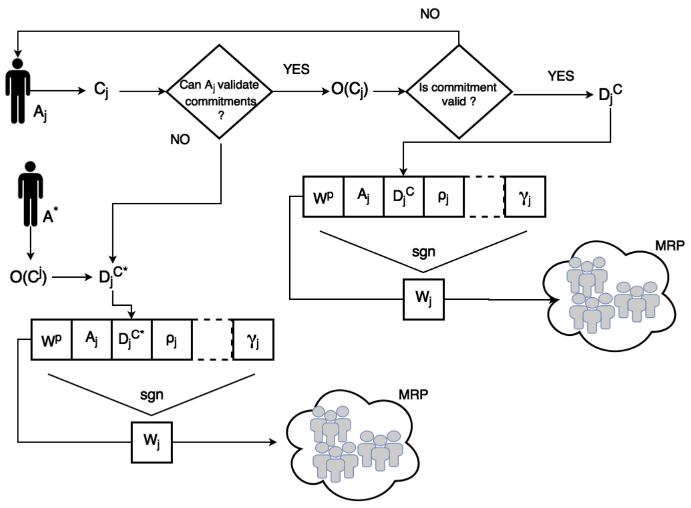

Figure 3: The proof of commitment workflow for agent A_j.

being temporally related to the issuing of the current block B;

2. this step forks in two cases depending if commitment verification is performed by A_j or it is performed by the super-agent $A^\star$:

   (a) A_j verifies the validity of C_j by evaluating $\mathsf{O}(C_j)$ and, if the above check is passed, then it computes the so called *commitment digest*, denoted as D_j^C, which uniquely represents with overwhelming probability C_j;

   (b) alternatively, A_j sends C_j via a secure channel to the super-agent $A^\star$, which verifies the validity of C_j and in case of a positive response returns to A_j the commitment digest $D_j^{C\star}$;

3. A_j composes its own current block header H_j which contains, among other fields, the witness W^p for the block header of the last valid block in the blockchain, its public identity A_j, the current commitment digest (given by D_j^C or $D_j^{C\star}$, depending on step 2), and the header's fields `tickets` and `issue` instantiated with the values ρ_j and γ_j, respectively. Then A_j computes its current witness W_j by signing H_j;

4. A_j is now ready to take part to the raffle protocol.

Each problem in the *Nondeterministic Polinomial time* or in the *Number P* complexity classes [25] is a possible problem P candidate. Indeed, problems in both these classes admit some efficient algorithm that, given any candidate solution for P, can verify if such candidate actually solves or not P.

Commitment digests D^C can play in this context two main roles, depending on the way the super-agent $A^\star$ intends to ascertain the work performed by agents on problem P. In case the commitment verification is left to agents A_i, commitment digests have no other scope than representing with a short encoding the commitment C_j, and it could be the case that $D_j^C = C_j$. Conversely, D_j^C must be an authentic token issued by $A^\star$ from which any other party different from $A^\star$ (and A_j) cannot disclose C_j. In this case, commitment digests can be obtained as suitable signatures computed by $A^\star$ on commitments.

Here witnesses W have the same function and processing than in the proof-of-transcription scheme. They can give evidence to receivers that a given agent performed a task without disclosing it, and they are the result of a signature algorithm `sgn` on a message obtained by collating various fields (possibly all) of a block header. If commitment verification is left to standard agents, it could be the case that A_j takes part to the current raffle by sending the couple (ρ_j, W_j) rather than (H_j, W_j). This way, at the cost of one more communication round for the raffle protocol, agents can keep secret their computations (or their candidate solution) until a check for the candidate winner is required. Other implementations are of course possible. For example, in the context of verifications performed without a super-agent, agents could protect their (eventually large) computations or solution sets through the digesting mechanism, by choosing as commitment digest the output of an hashing algorithm `hsh`.

5.3 The Raffle protocol

Our approach keeps the network operational through the work of a set of competing agents, and the raffle protocol is its core. Likewise miners in Bitcoin, in our approach agents are motivated to spend their effort to have a chance and win a reward. Unlike miners, however, winners are selected among a set of eligible agents in a way similar to what happens for the winner of a raffle. Moreover – as we have discussed in the preamble – whilst miners synchronize with each other on the basis of the average time they spend as a whole in doing their proof-of-work, agents can get in sync with respect to the various raffle extractions either with an implicit timing mechanism based on the tickets they own, or by relying on a trusted "global clock" service.

A key point in designing a peer-to-peer protocol for eliciting a winner among competing agents is *scalability*, and scalability turns out in a low number of explicit comparisons among the involved parties. In other words, each agent should be able to establish with adequate confidence if it could be or not the winner by performing only local computations. Since our approach would give rise ideally to systems that allow for verification of both transactions and commitments in real-time, our design aims to reduce communications among peers thanks to a *filtering mechanism* which scales down in a tunable fashion the number of agents that are eligible for the prize. The filtering mechanism is implemented through a *raffle function* Ω and a *threshold* τ. The raffle function Ω is an efficiently computable function with the following properties:

- Ω takes in input binary strings and returns positive numbers;

- A Ω evaluation can be computed privately by each agent A_j and it is uniquely tied to agent A_j through its current witness W_j;
- When agent A_j publishes its current couple (H_j, W_j), then any other agent can recompute the value obtained at the previous step by A_j for verification.

Two possible simple choices for Ω satisfying the above properties are:

$$\Omega_1(X) = (\mathtt{prg}(X))_2, \quad \Omega_2(X,Y) = |(\mathtt{hsh}(X)_2 - (\mathtt{hsh}(Y))_2|,$$

where `prg` and `hsh` respectively denote a suitable pseudo-random generator and a suitable hash algorithm, and X,Y are two input for those algorithms.
The threshold τ is a positive integer which sets a limit (above or below) for useful values obtained through the raffle execution:only agents that on the basis of their private computations of Ω satisfy the constraint imposed by τ are eligible for participating to the final comparison to select a unique winner. Filtering mechanisms related to the above choices for Ω are given for example by:

$$\begin{aligned} \max\{\Omega_1(W_j), \ldots, \Omega_1^{\rho_j - 1}(W_j)\} &> \tau \\ \min\{\Omega_1(W_j), \ldots, \Omega_1^{\rho_j - 1}(W_j)\} &< \tau \\ \min\{\Omega_2(W^p, W_j), \ldots, \Omega_2(W^p, \mathtt{hsh}^{\rho_j - 1}(W_j))\} &< \tau, \end{aligned}$$

where ρ_j is the number of tickets owned by A_j for the raffle execution, Ω_1^n is the nth functional power of Ω_1 and $\mathtt{alg}^n(X)$ denotes the iterative application of n runs of algorithm `alg` to input X.

In the following we sketch a possible raffle protocol under the assumption of the existence of a trusted global clock, e.g. in case of VC and VTC systems when the verification of commitments is performed by the super-agent, and by using Ω_2 and τ as described above. In this context, the *delay time* δ defines the minimum elapsed time any agent has to wait in order to proceed in processing information collected through the system network. It can be preset within the system on the basis of its network performance, and it can be publicly verified against the `issue` field recorded and signed in the messages exchanged among system nodes during a raffle's winner verification.

Raffle Execution To take part to the current raffle, A_j proceeds as follows.

E1 A_j uses the last witness W^p stored in the blockchain and its own current witness W_j in order to compute the ρ_j non negative integers $\Omega_2(W^p, W_j) \ldots \Omega_2(W^p, \mathtt{hsh}^{\rho_j - 1}(W_j))$, where ρ_j is the number of tickets it has earned for the current raffle;

E2 A_j computes its own draw value

$$\tau_j = \min\{\Omega_2(W^p, W_j) \ldots \Omega_2(W^p, \mathtt{hsh}^{\rho_j - 1}(W_j))\};$$

E3 iff $\tau_j \leq \tau$ then A_j broadcasts the couple (H_j, W_j) over the network for verification;

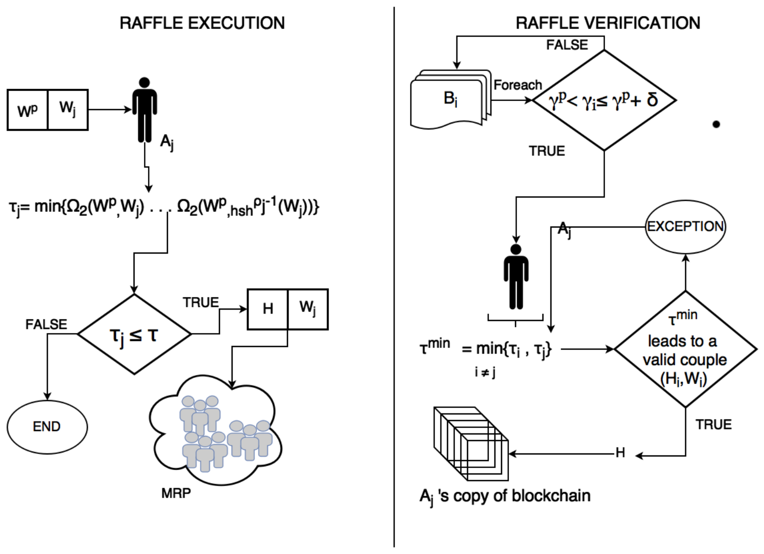

Figure 4: The raffle protocol workflow.

Raffle Verification To verify raffle executions by other agents, A_j proceeds as follows.

V1 For each current block B_i from A_i satisfying the time constraint imposed through the system delay time δ:

$$\gamma^p < \gamma_i \leq \gamma^p + \delta \ ,$$

where γ^p is the issue time of the last block stored in the blockchain, A_j does the computations at steps E1 and E2 using as input W_i and ρ_i and obtaining as output τ_i;

V2 A_j finds the minimum draw value $\tau^{\min}$ among the draw values τ_i computed at the previous step and its value τ_j;

V3 if $\tau^{\min}$ corresponds to a valid couple (H_i, W_i) from A_i, then A_j registers H_i as the new current block in its copy of the blockchain, otherwise A_j broadcasts over the network an exception towards A_i to blacklist it for the next raffle extraction, and performs the previous step again until a valid couple (H_i, W_i) is found;

Raffle Exit A_j is now ready to take part to the next raffle.

A winner agent is an agent satisfying the following requisites:

- it has issued a valid current block (in terms of its fields and their match with the witness);
- it has actually got the minimum riffle value;

- it did not receive any valid exception for the previous raffle;
- it signaled each cheating agent in the previous raffle, if any.

In general there could be multiple agents which have got the same minimum draw value $\tau^{\min}$ (which could possibly be zero), and in this case a unique winner can be selected on the basis of one or more additional criteria that are objectively verifiable, such as the highest number of owned tickets, the minimum value registered in the `issue` field of the block header, the highest witness (if viewed as a base 2 representation), and so on.

Our approach encompasses instances of the raffle protocol for which the selection of the winner does not require a majority consensus. That has the advantage of reducing drastically communications (step V1) and avoiding comparison at all (step V2) among agents, but at the cost of riffles without winners. For example, with reference to the raffle protocol sketched above, the threshold $\tau = 1$ would correspond to find a collision in the hashing algorithm `hsh`, an extremely rare event if `hsh` is a cryptographically strong hashing algorithm and the overall number of tickets owned by the competing agents is much less than $2^{|\mathtt{hsh}|/2}$, where $|\mathtt{hsh}|$ denotes the length of the hash digests returned by `hsh`. In such cases, an approach to counter the occurrence of "null" raffles and the related decay in system performance could be implemented through a super-agent which is used to store the prize for the next winner[7] and to add a block of its choice among a suitable set of valid couples (H, W).

6 Conclusions

In this work we pointed out the unsustainability of mining in case of massive large-scale blockchain-based system. We found that cryptocurrency systems like Bitcoin are potentially energy inefficient when projected to a global scale, and are prone to several risks of malicious control. Moreover the Nonce Race coupled with the consensus mechanism, as we also largely discussed in [1], entails serious drawbacks, and results ultimately in systems that are very far from achieving transaction validation in real-time.

Starting from such weakness, and after isolating the basic concepts behind mining, we sketched possible alternatives for the maintenance of blockchain-based systems. We considered security issues, incentives, as well as competitive and collaborative opportunities, and we proposed a framework of concepts and algorithms to implement blockchain-based systems for different contexts.

We expect to develop a prototype of a blockchain-based system with the raffle protocol in the near future.

[7]This results in a kind of raffle similar to the Italian game "Gratta e vinci".

References

[1] Pasquale Forte, Diego Romano, and Giovanni Schmid. Beyond bitcoin - part i: A critical look at blockchain-based systems. Cryptology ePrint Archive, Report 2015/1164, 2015. `http://eprint.iacr.org/`.

[2] Satoshi Nakamoto. Bitcoin: A peer to peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[3] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology - CRYPTO92*, pages 139–147. Springer, 1993.

[4] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, CMS '99, pages 258–272. Kluwer, B.V., 1999.

[5] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. `https://wallet.peercoin.net/assets/paper/peercoin-paper.pdf`, 2012.

[6] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoins proof of work via proof of stake. In *SIGMETRICS 2014 Workshop on Economics of Networked Systems, NetEcon*. ACM, 2014.

[7] Hass McCook. An order-of-magnitude estimate of the relative sustainability of the bitcoin network. `https://www.academia.edu/7666373`, 2015. Accessed: 2016-01-29.

[8] Jonathan G Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. Implications of historical trends in the electrical efficiency of computing. *Annals of the History of Computing, IEEE*, 33(3):46–54, 2011.

[9] Blockhain.info charts. `https://blockchain.info/charts/`. Accessed: 2016-02-11.

[10] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013. Citeseer, 2013.

[11] CoinDesk. State of bitcoin and blockchain 2016. `http://www.coindesk.com/research/state-bitcoin-blockchain-2016/`, 2015. Accessed: 2016-02-11.

[12] Bcg classics revisited: The rule of three and four. `https://www.bcgperspectives.com/content/articles/business_unit_strategy_the_rule_of_three_and_four_bcg_classics_revisited/`. Accessed: 2016-02-11.

[13] Adam Back. Hashcash – a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf, 2002. Accessed: 2016-05-15.

[14] Nick Szabo. Bit gold. `http://unenumerated.blogspot.it/2005/12/bit-gold.html`, 2008. Accessed: 2016-05-15.

[15] Joshua Kolden. Bitcoin@stack exchange: What are bitcoin “confirmations”? `http://bitcoin.stackexchange.com/questions/146/what-are-bitcoin-confirmations/160#160`. Accessed: 2015-11-29.

[16] 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.

[17] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2010.

[18] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption*, pages 371–388. Springer, 2004.

[19] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

[20] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[21] David Mills, Jim Martin, Jack Burbank, and William Kasch. Network time protocol version 4: Protocol and algorithms specification. *IETF RFC5905, June*, 2010.

[22] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in CryptologyCRYPTO87*, pages 369–378. Springer, 1988.

[23] Eric Bach and Jeffrey Outlaw Shallit. *Algorithmic Number Theory: Efficient Algorithms*, volume 1. MIT press, 1996.

[24] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[25] Laszlo Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 337–347. IEEE, 1986.