Distributed-Ledger Consensus Protocol for Digital Social Contracts

Ouri Poupko[1], Ehud Shapiro[1], and Nimrod Talmon[2]

[1] Weizmann Institute of Science, Rehovot, Israel
[2] Ben-Gurion University of the Negev, Beersheba, Israel

Abstract. Digital social contracts are the modern counterpart of Proudhon's anarchistic notion of social contracts – voluntary agreements among sovereign people (as opposed to Rousseau's, which justified state power). Digital social contracts are: Conceptually, voluntary agreements among genuinely-identified people, specified, undertaken, and fulfilled in the digital realm. Mathematically, a novel computational model specifying concurrent, non-deterministic asynchronous agents updating a monotonic shared ledger with digital speech acts. Computationally, distributed programs realizing this abstract model of computation, where people resolve nondeterministic choices of the agents they operate [2]. Here, we envision a setup that consists of people with genuine identifiers, which are unique and singular cryptographic key pairs, each operating a software agent thus identified on their mobile device, and address the distributed-ledger implementation of digital social contracts in the presence of faulty agents in this setup. We present a distributed-ledger transition system and show that it implements the abstract shared-ledger model of digital social contracts, and discuss its resilience to faulty agents. The final result is a novel ledger architecture that is truly distributed with a blockchain-per-person (as opposed to centralized with one blockchain for all), partially-ordered (as opposed to totally-ordered), locally-replicated (as opposed to globally-replicated), asynchronous (as opposed to globally-synchronized), peer-to-peer with each agent being both an actor and a validator (as opposed to having dedicated miners, validators, and clients), environmentally-friendly (as opposed to the environmentally-harmful Proof-of-Work), self-sufficient (as opposed to the energy-hogging Proof-of-Work or capital-hogging Proof-of-Stake) and egalitarian (as opposed to the plutocratic Proof-of-Work and Proof-of-Stake).

1 Introduction

A social contract is a voluntary agreement between people that is specified, undertaken, and fulfilled in the digital realm. It embodies the notion of "code-is-law"[8], in that a digital social contract is in fact a program – code in a social contracts programming language [2], which specifies the digital actions parties to the social contract may take; and the parties to the contract are entrusted, equally, with the task of ensuring that each party abides by the contract. Parties

to a social contract are identified via their public keys, and the one and only type of action a party to a digital social contract may take is a "digital speech act" – signing an utterance with her private key and sending it to the other parties to the contract.

Reference [2] presents a formal definition of a digital social contract as agents that communicate asynchronously via digital speech acts, where the output of each agent is the input of all the other agents. In particular, it offers an abstract design for a social contracts programming language and demonstrates, via programming examples, that key application areas, including social community; simple sharing-economy applications; egalitarian currency networks; and democratic community governance, can all be expressed elegantly and efficiently as digital social contracts.

Our goal here is to take the abstract notion of a digital social contracts [2] and describe a *fault-tolerant distributed ledger implementation* of such digital social contracts. To do so, we recall from [2] the definition of digital social contracts as an abstract transition systems. We then define incrementally a sequence of transition systems: First, a distributed implementation; then a strict fault-tolerant implementation with finalization [1]; then, a relaxed implementation, in which, similarly to blockchain protocols, agents may act based on non-final acts, but might have to abandon them if they are discovered later to be based on a double-act [3].

The final result is a novel blockchain architecture that is distributed with a blockchain-per-person, partially-ordered, locally-replicated, asynchronous, peer-to-peer, with each agent being both an actor and a validator, environmentally-friendly, and egalitarian, as summarized in Table 1.

While the aim of this paper is to formally define our fault-tolerant distributed ledger implementation in a mathematical way, we hint at certain possibilities of practical implementation of a system implementing the mathematical models developed herein, and also relate such an implementation to the programming language developed for digital social contracts.

1.1 Related Work

A fundamental tenet of our design is that social agreements are made between diverse, multi-faced, multi-player groups of people. This is in stark contrast to the design of cryptocurrencies and their associated smart contracts, which although they are distributed among multiple participants, and claim to run a distributed ledger, they actually run a single replicated ledger. A challenge that cryptocurrencies address is how to achieve consensus in the absence of trust, and their solution is based on proof-of-work [4] or, more recently, proof-of-stake [10] protocols. In contrast, social contracts are between individuals, each expected to posses a genuine (unique and singular) identifier [13] (see therein discussion on how this can be ensured).

Interaction between genuine digital identities is susceptible to sybils, which in this case are either fake or duplicate. A prior line of work [11] shows how a community of individuals with genuine identifiers can maintain a bound on

	Standard Blockchain	Digital Social Contracts
1. Central vs. Distributed	One blockchain for all	A blockchain per person
2. Totally- vs. Partially-Ordered	Linear global history	Partial ordering among personal histories
3. Global vs. Local	Actions replicated globally	Local replication among parties to a contract
4. Synchronous vs. Asynchronous	Actions synchronized globally by a single (rotating) miner	Each party synchronizes actions independently
5. Client-Server vs. Peer-to-Peer	Miners vs. validators vs. clients	All are equal as actors and validators (and minters)
6. Resource-Hogging vs. Frugal	Energy-hogging (Proof-of-Work) or capital-hogging (Proof-of-Stake)	Self-Sufficient
7. Plutocratic vs. Egalitarian	Proof-of-Work/ Proof-of-Stake	Egalitarian control

Table 1. Blockchain Architectures

the ratio of sybils. Additional work [6] shows how a cryptocurrency can further incentivize eradicating sybils and deter sybil creation. The approach taken in this paper assumes some number of sybils may still exist in the community, yet the sybils and their corrupt operators are still within the limits of the byzantines bound required for maintaining fault tolerance.

Assuming genuine identities, with not too many sybils, a different approach can be taken, other than achieving consensus. In our approach, the integrity of the record of actions taken by the parties to the social agreement is reached between parties to the agreement, not between external anonymous "miners", as in cryptocurrencies. This gives rise to a much simpler approach to fault tolerance.

In particular, our approach does not suffer from forks as for example the Bitcoin protocol [9], and does not need to reach Byzantine Agreement [7]. Instead, agents ratify actions of each other, an action is decided to be final when it is known that a supermajority of the agents ratify it, and agents may take an action that depends on actions of others only once they are decided to be final. As a result, the handling of "double spend", a key challenge for cryptocurrencies, is much simpler.

Hence, rather than building on an existing blockchain architecture, we propose here a new one. Digital social contracts are (also) an abstract model of computation, and as such could be implemented on any Turing-complete computational model, in particular on a single centralized computer, on a client-server system, as a software-as-a-service application, as a smart contract on a public (permissionless) blockchain, or on a permissioned blockchain. Digital social contracts aim to support digital communities that are both sovereign and egalitar-

ian: sovereign over their membership and conduct, and egalitarian in exercising their sovereignty. Open peer-to-peer systems, e.g., Scuttlebut [14], do not exercise sovereignty over membership; federated systems, e.g.. Mastodon [12], do not exercise egalitarian governance, in that each server operator has full control over the community residing on its server. Equality without sovereignty is easily achieved, in principle, among the clients in a client-server architecture. But democratic governance of a system is impossible if the members are not the sovereign, as any decision they make can ultimately be overruled by the sovereign, who can simply unplug/erase/block the community. Blockchain is the first technology to offer sovereignty to its participants: Nobody can unplug Bitcoin or Ethereum, for example; both will keep running as long as someone somewhere keeps running their protocols. However, present public (permissionless) blockchains are not a feasible vehicle for digital social contracts, for several reasons. First, their Proof-of-Work protocol is environmentally harmful, and hence cannot be supported by any person of conscience, let alone an effort aiming at a just society. Second, they, as well as novel environmentally-friendly consensus protocols [10,5], are plutocratic, providing more power and more wealth to those who are already wealthy; hence they are neither egalitarian nor just. Standard private (permissioned) dedicated blockchains can address the challenge of sovereignty and, possibly with major extra effort, could be made egalitarian and just. However, the standard blockchain architecture has one global blockchain shared by all agents, which stores all actions by all agents in a single linear data structure; the entire blockchain is fully-replicated to all agents, and all actions of all agents are synchronized by a single (rotating) agent. These aspects hamper the scalability of the standard blockchain architecture. Hence we aim for a novel architectural foundations.

2 Construction Outline

A preprint of the full paper is available (arXiv:2006.01029). It presents the entire rather lengthy construction, including proofs. An outline of the construction and the main ideas in it are presented in Figure 1.

The construction starts with a distributed implementation of a ledger. As each agent has one identifier, and each agent is a minter, the construction distributes the ledger between participants, such that each one controls and manages only its personal linearized history of speech acts. This includes both its own output messages, as well as messages it inputs from other agents. Full linearization between agents is not required, and that is the main difference from existing permissioned blockchains. The paper proves that this construction implements correctly a transition system of a single, shared, ledger.

The second step of the construction introduces an adversary model that is the same as conventional existing permissioned consensus protocols. It uses ratification messages between the agents, similar to Bracha [1], but since each agent maintains its own line of history, it does not need to wait until consensus is

Section 3: SC – shared ledger of personal histories
- Acts are signed
- Input and output acts access shared ledger

Section 4: DSC – distributed ledger of personal histories, asynchronous network
- Input and output acts access personal history and network

Section 5: FTDSC – fault-tolerant distributed ledger, bags of meta-messages
- Acts are indexed, include preceding inputs, and then signed
- Agents ratify acts of others based on knowledge of their history and finality
- Act is final if ratified by a δ-supermajority of the agents
- Input act can be taken if act is final

Section 6: RFTDSC – relaxed fault-tolerant distributed ledger
- Input acts can be taken even if not final
- If a double *u*-act is discovered then *u* is declared faulty
- Non-final *u*-acts can then be rejected by a δ-superminority
- Acts that depend on rejected *u*-acts are regretted

Section 7: CRFTDSC – chained relaxed fault-tolerant distributed ledger
- Personal histories are blokchains
- Acts include hash pointer to history and succinctly reference preceding acts
- Integrity of acts can be validated using hash pointers

Fig. 1. Outline of Paper

reached. Rather, once a δ-supermajority of ratification messages is recieved, assuming 2δ faulty agents, the agent can continue and execute further transactions.

A fundamental leniency in digital social contracts, compared to standard blockchains, is that once an agent is determined to be faulty, its faulty actions and any subsequent actions of it can be ignored. In particular, if a double-act is detected before neither of its two branches has been finalized, both acts could be rejected. We combine this leniency with the following protocol idea: Once an agent knows that another agent has performed a faulty act, the first agent undertakes to never contribute to finalizing any of the second agent's acts, including and following the faulty act, and informs all other agents of this undertaking. Hence, once an agent knows that a δ-supermajority of the agents intend to reject an act, it may conclude that it can never be finalized by a δ-supermajority, and hence may ignore it and any subsequent acts on the branch. In particular, if agents act optimistically and proceed based on non-final acts, they may have to roll-back in case such an act is determined to be part of a double-act before being finalized. The result could be a rather efficient protocol, where agents proceed based on trusting other agents for sufficiently-small transactions, base on their level of trust of the other agents, and waiting for finalization in case of rather large transactions or transactions with agents they have less reasons to trust.

This, combined with the properties of our distributed-ledger protocol, may result in a highly-efficient and scalable fault-tolerant distributed foundation for digital social contracts.

Acknowledgements

We thank Luca Cardelli for lengthy discussions and excellent advice. Ehud Shapiro is the Incumbent of The Harry Weinrebe Professorial Chair of Computer Science and Biology. We thank the generous support of the Braginsky Center for the Interface between Science and the Humanities. Nimrod Talmon was supported by the Israel Science Foundation (ISF; Grant No. 630/19).

References

1. Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.
2. Luca Cardelli, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Digital social contracts: A foundation for an egalitarian and just digital society, 2020.
3. Usman W Chohan. The double spending problem and cryptocurrencies. *Available at SSRN 3090174*, 2017.
4. Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.
5. Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017.
6. Avigail Gurin-Schleifer, Ouri Poupko, Ehud Shapiro, and Nimrod Talmon. Sybil-resilient, egalitarian and just digital currency, 2020.
7. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem, 2019.
8. Lawrence Lessig. *Code Is Law – On Liberty in Cyberspace*. Harvard Magazine, 2000.
9. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2019.
10. Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.
11. Ouri Poupko, Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Sybil-resilient conductance-based community growth. *CoRR*, abs/1901.00752, 2019.
12. Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the decentralised web: The mastodon case. In *Proceedings of the Internet Measurement Conference*, pages 217–229, 2019.
13. Gal Shahaf, Ehud Shapiro, and Nimrod Talmon. Foundation for genuine global identities. *arXiv preprint arXiv:1904.09630*, 2019.
14. Dominic Tarr, Erick Lavoie, Aljoscha Meyer, and Christian Tschudin. Secure scuttlebutt: An identity-centric protocol for subjective and decentralized applications. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, pages 1–11, 2019.