Building cryptotokens based on permissioned blockchain framework

Oleksandr Anyshchenko[1], Ivan Bohuslavskyi[1], Stanislav Kruglik[2], Yash Madhwal[2], Alex Ostrovsky[1] and Yury Yanovich[1,2,3]

Abstract— Blockchain technology has a lot of applications including but not limited to cryptotokens. For example, permissioned blockchains for state registries can include no money logic inside. But the creation of cryptotoken is a typical example of blockchain application and is a popular model to measure performance. In this paper, we described how to construct an account-based cryptotoken using Exonum, an open-source framework for creating blockchain applications, and by the means of performance tests shown that the proposed solution meets real-life throughput requirements for many applications. The proposed approach can be used to create cryptocurrencies in other open-source frameworks focused on permissioned blockchain applications.

I. INTRODUCTION

Since Bitcoin [1], blockchains took a prominent place in the field of distributed networks. And subsequently they found applications in many areas (state registers, supply-chain management, biomedicine, finance, etc. [2], [3], [4], [5], [6], [7], [8], [9], [10]). Recently blockchain technology has found its' application in wireless networks management especially in the next generation vehicular communication [11], [12], [13]. Blockchains could be categorized by the level of access to the data on public and private ones with several extra subdivisions [14], [15], [16], [17]. Each of these types has its own application area and limitations.

Private blockchains can contain no token or cryptocurrency inside. Government registries are typical examples of such systems. Nevertheless, the cryptotoken is a typical example of blockchain applications with constantly growing amount of consumers [18]. In this paper, we propose an approach to build cryptotokens based on permissioned blockchain framework and test in on Exonum [19] showing that it meets real-world requirements by means of performance tests. Namely, we create and test service on Exonum that implements a cryptocurrency, create cryptographic proofs for data and show how to organize the corresponding data layout. This approach can be used in other open-source blockchain platforms which don't basically intend to cryptocurrency creation.

The rest of the paper is organized as follows:

- Exonum overview is provided in Section II.
- Cryptocurrency description and implementation details are in Section III.
- Preformance tests and their results are provided in Section IV.

II. EXONUM OVERVIEW

Exonum is an extensible open-source framework for creating blockchain applications that can be used to create cryptographically powered distributed ledgers in virtually any problem domain (https://exonum.com/doc/architecture). It is oriented towards creating private and public permissioned blockchains, that is, blockchains with the known set of blockchain infrastructure providers.

A. Services

Services are the main extension point for the Exonum framework. By itself, Exonum provides building blocks for creating blockchains with any concrete transaction processing rules. Its' services encapsulate business logic of the blockchain application.

- A service specifies the rules of transaction processing, namely, how transactions influence the state of the service.
- The state transformed by transactions is persisted as a part of the overall blockchain key-value storage.

The authors are listed in alphabetical order.

The research of Stanislav Kruglik was supported by RFBR according to the research projects no. 18-07-01427, 18-37-00459, 19-01-00364, 19-37-80006.

The authors would like to express their most profound appreciation to Exonum team.

[1] Software Department, Bitfury, 1016 BP Amsterdam, The Netherlands

[2] Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and Technology (Skoltech), 121205 Moscow, Russia

[3] Laboratory of Data Mining and Predictive Modeling, Institute for Information Transmission Problems (IITP RAS), 127051 Moscow, Russia

- A service may also allow external clients to read the relevant data from the blockchain state.

Each service has a well-defined interface for communication with the external world essentially, a set of endpoints and the implementation of said interface. The implementation may read and write data from the blockchain state (usually using the schema helper for the underlying key-value storage in order to simplify data management) and can also access the local node configuration.

Services are executed on each validator and each auditing node of the blockchain network. The order of transaction processing and the resulting changes to the service state are a part of the consensus algorithm. They are guaranteed to be the same for all nodes in the blockchain network.

In order to communicate with external entities, services employ three kinds of endpoints:

- Transactions.
- Read requests (together with transactions, form public application programming interface (API)).
- Private API.
- Service endpoints are automatically aggregated and dispatched by the Exonum middleware layer.

B. Transactions

A transaction in Exonum, as in usual databases, is a group of sequential operations with the data (i.e., the Exonum key-value storage). Transaction processing rules are defined in services; these rules determine business logic of any Exonum-powered blockchain.

Transactions are executed atomically, consistently, in isolation and durable. If the transaction execution violates certain data invariants, the transaction is completely rolled back, so that it does not have any effect on the persistent storage.

If the transaction is correct, it can be committed, i.e., included into a block via the consensus algorithm among the blockchain validators. Consensus provides total ordering among all transactions; between any two transactions in the blockchain, it is possible to determine which one comes first. Transactions are applied to the Exonum key-value storage sequentially in the same order transactions are placed into the blockchain.

All transactions are authenticated with the help of public-key digital signatures. Generally, a transaction contains the signature verification key (aka public key) among its parameters. Thus, authorization (verifying whether the transaction author actually has the right to perform the transaction) can be accomplished with the help of building a public key infrastructure and/or various constraints based on this key.

C. System Configuration

System configuration is a set of parameters that determine the network access parameters of a node and behavior of the node while operating in the network.

The configuration is stored in the TOML format (https://github.com/toml-lang/toml). A path to the configuration file should be specified on the node startup.

The configuration may be changed using the global variables updater service or by editing the configuration file.

Services may have their own configuration settings. On node initialization, the configuration is passed to all services deployed in the blockchain. The configuration for a service is stored in the services configs subtree of the overall configuration under a separate key equal to the name of the service.

D. Data Model

The schema is a structured view of the key-value storage used in Exonum. The abstractions that are used in client applications are close to RocksDB (https://rocksdb.org/) database engine:

1) Exonum table types lists supported types of data storage collections. Tables represent the highest abstraction level for data storage.
2) Low-level storage explains how tables are persisted using RocksDB.
3) View layer describes the wrapper over the DB engine that ensures atomicity of blocks and transactions.
4) List of system tables contains tables used directly by the Exonum core.
5) Indexing gives an insight how indices over structured data can be built in Exonum.

To access the storage, however, we will not use the storage directly, but rather Snapshots and Forks. Snapshot represents an immutable view of the storage, and Fork is a mutable one, where the changes can be easily rolled back. Snapshot is used in read requests, and Fork in transaction processing.

E. Serialization Format

Binary serialization format is used in Exonum for communication among full nodes, cryptographic operations on light clients and storage of data. The format design provides several important properties, including resilience to maliciously crafted messages, zero-copy deserialization and canonicity.

Serialization in Exonum differs from serialization in the usual sense, since there is no process of transforming the structure into binary data. The data is created already "serialized" and Exonum works directly with the serialized data "deserializing" the fields to which it refers, if necessary.

F. Consensus

Generally, a consensus algorithm is a process of obtaining an agreed result by a group of participants [20], [21]. In Exonum the consensus algorithm is used to agree on the list of transactions in blocks added to the blockchain. The other goal of the algorithm is to ensure that the results of the transaction execution are interpreted in the same way by all nodes in the blockchain network.

The consensus algorithm in Exonum uses some ideas from the algorithm proposed in Tendermint [22], but has several distinguishing characteristics as compared to it and other consensus algorithms [23], [16] for blockchains.

III. CRYPTOCURRENCY IMPLEMENTATION

In this paper we implement a cryptocurrency, which allows the following operations:

- Creating a wallet.
- Replenishing the wallet balance.
- Transferring money between wallets.
- Obtaining cryptographic proofs of executed transactions.
- Reviewing wallets history.

You can view and download the full source code of this example from https://github.com/exonum/exonum/tree/master/examples/cryptocurrency-advanced. Below we briefly explain the main realization steps.

A. Declare Persistent Data

We should declare what kind of data the service will store in the blockchain. In our case we need to declare a single type — `Wallet`. Inside the wallet we want to store:

- Public key to validate requests from the owner of the wallet
- Name of the owner (purely for convenience reasons)
- Current `balance` of the wallet
- The length of the wallet history and its hash.

B. Define Transactions

In this paper we use the following service transactions:

- `CreateWallet` to create a new wallet
- `Transfer` money between two different wallets
- `Issue` transaction issue new founds. It also contains a seed to avoid replay of the transaction.

C. Reporting Errors

The execution of the transaction may be unsuccessful for some reason. For example, the creating new wallet will not be executed if the wallet with such public key already exists. There are also three reasons why the transaction of money transfer cannot be executed:

- There is no sender with a given public key
- There is no recipient with a given public key
- The sender has insufficient currency amount.

D. Transaction Execution

Every transaction in Exonum has business logic of the blockchain attached, which is encapsulated in the `Transaction` trait. This trait includes the verify method to verify the integrity of the transaction, and the `execute` method which contains logic applied to the storage when a transaction is executed.

IV. EXPERIMENTS

A. Environment Design

We performed experiments with two network configurations: in a single data center (DC) and multiple geographically distributed DCs. In both cases, twenty-one virtual machines were used (16 validators, 4 transaction generators, 1 benchmark control instance). Each validator was running on a separate virtual machine with 3.75 GiB RAM, 2 Core Intel Xeon Platinum CPUs running @3.4GHz, and the blockchain database was stored on an EBS drive connected to each instance. Nodes used Exonum version 0.9. In case of

- **Single DC:** virtual machines were in one availability zone within one Amazon Web Services (AWS) region (EU-central-1).
- **Multiple DCs:** validators were distributed among 14 AWS DCs in different locations: N. Virginia, Ohio, N. California, Oregon, Mumbai, Seoul, Singapore, Sydney, Tokyo, Canada, Frankfurt, Ireland, London and Paris. Generators were distributed in 4 different DCs in different regions: N. Virginia, Sydney, Tokyo and Frankfurt. Benchmark control instance was located at the Frankfurt AWS DC.

B. Blockchain Design

In all experiments the following consensus parameters were used

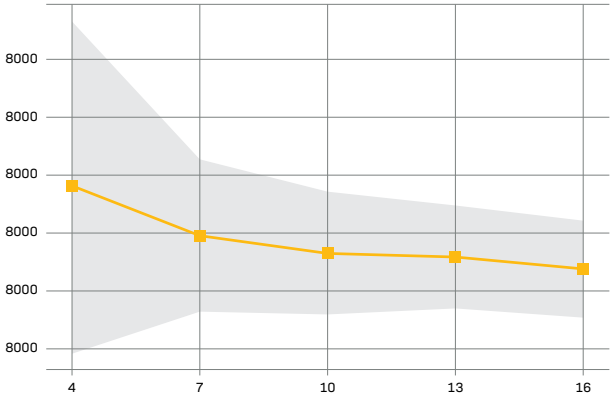

Fig. 1. **TPS** as a function of validators number in a **Single DC**. The black line with circles and green filling represents the mean and standard deviation for the cryptocurrency. Note that therein and after the Y axis corresponds to TPS.

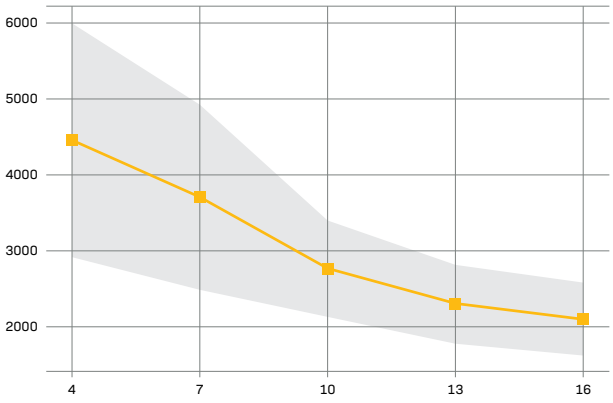

Fig. 2. **TPS** as a function of validators number in **Multiple DCs**.

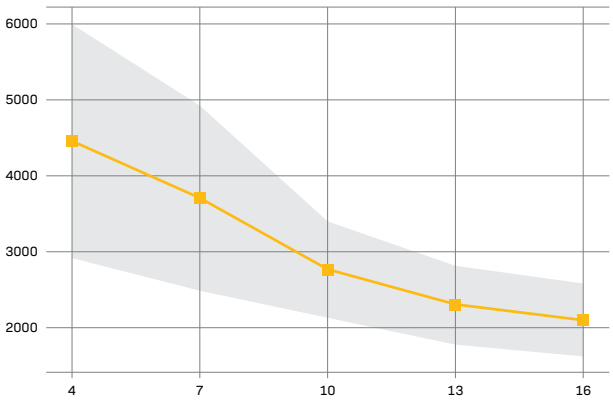

Fig. 3. **TPS** as a function of working validators fraction in a **Single DC**. The black line with circles and green filling represents the mean and standard deviation for the cryptocurrency.

- block capacity: 2000 transactions
- propose timeout: 0 seconds
- signature size: 64 bytes.

Note that propose timeout is an Exonum blockchain parameter. It is a delay before a block proposal for a new height. Its positive values are

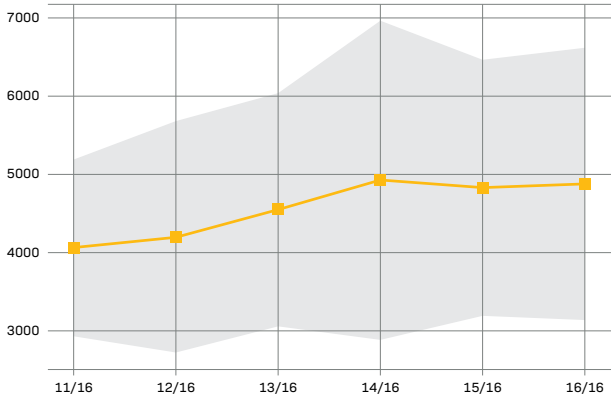

Fig. 4. **TPS** as a function of working validators fraction in **Multiple DCs**.

useful for networks with delay. In the paper the experiment was performed in a single data center. So we set the parameter value equal to zero. Four generators send transactions to all validators during the experiments with a constant flow. The flow is chosen in order to be a bit bigger than the blockchain TPS. Each validator check transaction signature before adding a transaction into the pool of unconfirmed transaction. Given scenario can be considered as a real-life high-load mode.

C. Performance Tests

We measured transactions per second (TPS, the bigger the better) for the different total amount of validators with fixed maximum block size of 2,000 transactions. The results were averaged over 100,000 transactions processed for each case. Hereafter the black line with circles and green filling represents the mean and standard deviation for the cryptocurrency.

Note. Almost all the blocks were filled with transactions

in our experiments, so one can estimate block acceptance time in seconds as 2000 TPS.

1) **Different Validators Number:** The number of consensus messages over network grows as a square of validators number in Exonum. It decreases blockchain performance. We considered different number of validators to estimate this effect.

TPS experimental results are in Figure 3. for a **Single DC** and in Figure 4 for **Multiple DCs**.

Exonum shows around 5000 TPS in a **Single DC**. This amount is almost independent of validators number as most likely writing Merkle proofs to EBS Drive process is the most time-consuming.

Exonum shows more than 4000 TPS in the **Multiple DC** for the cryptocurrency with 4 validators and more than 2000 TPS for 16 validators.

2) **Fail-Stop Validators:** The simplest model of validators failures is fail-stop. We choose it to demonstrate how improper nodes behavior slows consensus down. The total number of validators was 16 and from 0 to 5 were stopped (up to 1/3). TPS as a function of working validators fraction is in Figures 5 and 6.

Exonum shows more than 4000 TPS in a **Single DC** and more than 1800 TPS in **Multiple DCs**. This amount decreases with the working validators fraction decrease up to 20% maximum value.

V. CONCLUSION

We have described how to construct an account-based cryptotoken using Exonum. The token logic was implemented as an Exonum service. Although currently the service is just a proof of concept, it showcases how the Exonum platform can be used to build complex blockchain application. The performance tests show that the proposed solution meets real-life throughput requirements for many subject areas, including electronic payment systems. This solution can also be extended to other blockchain frameworks.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," www.bitcoin.org, pp. 1–9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] M. Swan, "Summary for Policymakers," in Climate Change 2013—The Physical Science Basis, Intergovernmental Panel on Climate Change, Ed. Cambridge: Cambridge University Press, 2015, pp. 1–30.

[3] M. Pilkington, "Blockchain Technology: Principles and Applications", in Research Handbook on Digital Transformations. Springer, 2016, pp. 225–253.

[4] H. M. Kim and M. Laskowski, "Towards an Ontology-Driven Blockchain Design for Supply Chain Provenance", SSRN Electronic Journal, vol. 25, No. 1, pp. 18–27, 8 2016. [Online]. Available: http://www.ssrn.com/abstract=2828369.

[5] T.-T. Kuo, H.-E. Kim, and L. Ohno-Machado, "Blockchain distributed ledger technologies for biomedical and health care applications", Journal of the American Medical Informatics Association, vol. 24, No. 6, pp. 1211–1220, 11 2017.

[6] S. Angraal, H. M. Krumholz, and W. L. Schulz, "Blockchain Technology," Circulation: Cardiovascular Quality and Outcomes, vol. 10, No. 9, pp. 5665–5690, 9 2017.

[7] P. Mamoshina, L. Ojomoko, Y. Yanovich, A. Ostrovski, A. Botezatu, P. Prikhodko, E. Izumchenko, A. Aliper, K. Romantsov, A. Zhebrak, I. O. Ogu, and A. Zhavoronkov, "Converging blockchain and next-generation artificial intelligence technologies to decentralize and accelerate biomedical research and healthcare", Oncotarget, vol. 9, No. 5, pp. 5665–5690, 1 2018. [Online]. Available: http://www.oncotarget.com/fulltext/22345.

[8] Y. Yanovich, I. Shiyanov, T. Myaldzin, I. Prokhorov, D. Korepanova, and S. Vorobyov, "Blockchain-Based Supply Chain for Postage Stamps", Informatics, Vol. 5, No. 4, p. 42, 11 2018.

[9] D. Korepanova, S. Kruglik, Y. Madhwal, T. Myaldzin, I. Prokhorov, I. Shiyanov, S. Vorobyov, and Y. Yanovich, "Blockchain-based solution to prevent postage stamps fraud", in 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), May 2019, pp. 171–175.

[10] N. Alzahrani and N. Bulusu, "Block-Supply Chain: A New Anti-Counterfeiting Supply Chain Using NFC and Blockchain," in Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems—CryBlock'18. New York, New York, USA: ACM Press, 2018, pp. 30–35.

[11] V. Ortega, F. Bouchmal, and J. F. Monserrat, "Trusted 5g vehicular networks: Blockchains and content-centric networking," IEEE Vehicular Technology Magazine, vol. 13, No. 2, pp. 121–127, June 2018.

[12] P. C. Bartolomeu and J. Ferreira, "Blockchain enabled vehicular communications: Fad or future?" in 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Aug 2018, pp. 1–5.

[13] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory", IEEE Transactions on Vehicular Technology, Vol. 68, No. 3, pp. 2906–2920, March 2019.

[14] Bitfury Group and J. Garzik, "Public versus Private Blockchains. Part 1: Permissioned

Blockchains", bitfury.com, pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf.

[15] ——, "Public versus Private Blockchains Part 2: Permissionless Blockchains", bitfury.com, pp. 1–20, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt2-1.pdf.

[16] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric", IBM Research, July, 2016.

[17] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends", in 2017 IEEE International Congress on Big Data (BigData Congress). IEEE, 6 2017, pp. 557–564. [Online]. Available: http://ieeexplore.ieee.org/document/8029379/.

[18] M. Szmigiera, "Number of Blockchain wallet users worldwide from 2nd quarter 2016 to 2nd quarter 2019", 2019. [Online]. Available: https://www.statista.com/statistics/647374/worldwide-blockchain-wallet-users/.

[19] Y. Yanovich, I. Ivashchenko, A. Ostrovsky, A. Shevchenko, and A. Sidorov, "Exonum: Byzantine fault tolerant protocol for blockchains", 2018.

[20] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony", Journal of the ACM, vol. 35, No. 2, pp. 288–323, 4 1988. [Online]. Available: http://portal.acm.org/citation.cfm?doid=42282.42283.

[21] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, pp. 382–401, 7 1982. [Online]. Available: http://portal.acm.org/citation.cfm?doid=357172.357176.

[22] J. Kwon, "TenderMint: Consensus without Mining", pp. 1–10, 2014. [Online]. Available: tendermint.com/docs/tendermint.pdf.

[23] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The Honey Badger of BFT Protocols", in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security — CCS'16. New York, New York, USA: ACM Press, 2016, pp. 31–42. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2976749.2978399, https://eprint.iacr.org/2016/199.pdf.