The Ritva Blockchain: Enabling Confidential Transactions at Scale

Henri Aare, Peter Vitols
Crystal Technology Research

May, 2020

Abstract

The distributed ledger technology has been widely hailed as the breakthrough technology. It has realised a great number of application scenarios, and improved workflow of many domains. Nonetheless, there remain a few major concerns in adopting and deploying the distributed ledger technology at scale. In this white paper, we tackle two of them, namely the throughput scalability and confidentiality protection for transactions. We learn from the existing body of research, and build a scale-out blockchain platform that champions privacy called RVChain. RVChain takes advantage of trusted execution environment to offer confidentiality protection for transactions, and scale the throughput of the network in proportion with the number of network participants by supporting parallel shadow chains.

1 The current state of affair

The distributed ledger technology (DLT), or commonly referred to as blockchain technology, provides a transparent and secure manner to record transactions and manage assets. The transparency comes from the fact that the ledger can be made public, and thus it is subject to universal audit. The security is guaranteed thanks to cryptographic mechanisms that make transactions, once recorded on the blockchain, are irreversible. These very properties enable DLT to attract a significant amount of attention. The technology has gained enormous traction since the birth of Bitcoin (BTC) [27]. The two most popular blockchain networks at this time of writing are Bitcoin and Ethereum [9] networks. These two blockchains together manage hundreds of billions of dollars in assets. As impressive as this number appears, it only touches a very small fraction of the amount of assets currently circulated in the market. There remains a gigantic room for improvement, expansion, and adoption. We pay our primary attention to privacy and processing throughput of the transactions.

The first hindrance that deters a wide adoption of blockchain technology in our everyday life is its limited scalability [8, 35, 18, 24]. While conventional centralized brokers such as VISA and PayPal can process thousands of transactions

per second, the Bitcoin network only supports 5-8 transactions per second [15]. Ethereum, which is proclaimed to be the general-purpose smart-contract enabled blockchain platform, is only able to handle approximately 20 transactions per second [34, 8]. Another factor that is worth considering is the non-finality and confirmation latency. A transaction on the Bitcoin network takes up to an hour to be reliably confirmed (i.e., the transaction's relevant parties are confident that it will not be reversed), whereas that number for the Ethereum network is about 10 minutes [1]. These weaknesses pose too much of a burden on any large-scale financial service.

There is yet another concern that Bitcoin and Ethereum networks admit, which is its lack of privacy/confidentiality protections for transactions. All transactions and/or data posted on these blockchains are not only visible to their relevant parties, but also to the public. That is, they cannot safely store or compute on sensitive data (e.g., clinical record, financial transactions) [11]. While both Bitcoin and Ethereum networks adopt pseudonyms so as to offer a certain level of privacy protections, a large body of research has shown that de-anonymizing such pseudonyms is feasible [29, 21].

2 Bridging the gap with the RVChain

We at Ritva set out to mitigate the aforementioned current state of affair by developing a performative general-purpose blockchain platform that supports confidential transactions, called RVChain. Abstractly, The RVChain takes advantage of the recent development in computer hardware, in particular CPUs that are capable of provisioning Trusted Execution Environment (TEE) [23, 4, 3, 33, 6, 5, 12]. Another technical feature that enables the RVChain to operate at scale is its capability to support parallel shadow chains whose transactions can be totally ordered [34].

By incorporating TEE, RVChain effectively simplifies the threat model it has to deal with to crash fault tolerance [20, 10, 7, 26], to which a number of performative consensus protocols have been studied. To further scale the transaction throughput of the platform alongside with its network size, RVChain allows virtually unlimited number of shadow chains (subject only to the network size) to complement one another. Network participants (or miners) are assigned to a chain uniformly at random, and leverage crash fault tolerance consensus protocol to establish a total ordering of transactions on that chain. RVChain then relies on a simple solution to establish a global order for transactions across the parallel shadow chains, thereby attaining consistency. The preliminary design of the RVChain currently supports approximately 2500 transactions per second for the network consisting of 100 participants. Its theoretical foundation allows the transaction to be processed at the network speed. That is, the only limit to the performance of the RVChain is the speed at which it receives transactions. With 5G technology on the horizon, the capacity is virtually limitless.

The other advantage of the TEE is its isolated execution [23]. More specifically, the TEE provisions a protected address space. Code and data running

and being processed in this protected address space is inaccessible to any unauthorised processes, even the privileged ones such as the Operating System or Hypervisor. When there is a need to write the sensitive data off the protected address space to the secondary memory, the TEE architecture transparently encrypts the sensitive data with cryptographic keys only available to the processor. RVChain makes use of this isolated execution feature offered by the TEE to provide stronger privacy and confidentiality protections for the sensitive transactions. In particular, the sensitive transactions and their data are stored encrypted on the public ledger. When they need to be processed, they are loaded into the protected address space of the TEE, decrypted and consumed therein. The output or updated state of the data and/or transactions are encrypted before being written off the protected address space to the public ledger. Consequently, only ciphertext (i.e., encrypted form) of the sensitive transactions and their data are visible publicly, while their integrity and consistency are accounted for by the TEE and the consensus protocol in use. Thanks to the semantic security of the encryption scheme in use, little, if not none, information can be inferred from the ciphertexts, hence the confidentiality.

3 Technical Foundations

3.1 Trusted Execution Environment

Intel recently proposed a set of CPUs that are capable of provisioning TEE, in particular Intel SGX. It enables a host to instantiate one or multiple TEEs, or enclaves, simultaneously. An enclave is associated with a CPU-guarded address space which is accessible only by the enclave code; the CPU blocks any non-enclave code's attempt to access the enclave memory. This effectively isolates the enclave from other enclaves concurrently running on the same host, from the OS, and from other user processes, thereby providing confidentiality and integrity protections for data and code loaded inside the enclaves. Memory pages can be swapped out of the enclave memory, but they are encrypted using the processor's key prior to leaving the enclave. Enclaves cannot directly execute OS-provided services such as I/O. In order to access those services, enclaves have to employ OCalls (calls executed by the enclave code to transfer the control to non-enclave code) and ECalls (API for untrusted applications to transfer control back to the enclave). These ECalls and OCalls constitute the enclave boundary interface, enabling a communication between the enclave code and the untrusted application to service OS provided functions. The RVChain leverages Intel SGX to implement the TEEs.

3.2 Crash Fault Tolerance Consensus Protocol

A blockchain is essentially a ledger maintained by independent nodes (servers, processors) that are geographically distributed. The consistency of the blockchain relies on these nodes reaching a consensus. A large body of research has been

dedicated to consensus protocols, addressing this problem. Consensus protocols primarily tackle either Byzantine failure model or crash failure model [13]. In this paper, we pay our attention to a consensus protocol that is designed for the crash failure model [19]. As the name suggests, this threat model assumes that a faulty node crashes (i.e., become indefinitely unresponsive), but never deviates from its intended behaviour. Our solution builds on the Raft consensus protocol [28] for its simplicity.

The protocol [28] is designed for a network of n deterministic nodes. The maximum number of faulty nodes the protocol can tolerate is $f = \frac{n-1}{2}$. Each node maintain a log which records an ordered list of transactions. The protocol guarantees that logs of non-faulty nodes converge. That is, they record the same sequence of transactions. A process can assume one of the following three roles, which are follower, candidate and leader.

Time is split into terms that are numbered with consecutive integers. In each term, there is one node being elected as the leader, while the remaining nodes serve as followers. The leader keeps its authority by exchanging heartbeat messages with all the followers periodically. Should a follower fail to receive any message from the leader after an election time-out has expired, it considers the leader as having been crashed. It then increments its term value, switches its role to being candidate, and requests for votes to become the new leader. The candidate obtains the leadership if it manages to obtains votes from a majority of other nodes in the network.

In normal operation, the followers respond only to messages and requests they receive from the leader and candidate, remaining passive otherwise. All the transactions (e.g., commands or requests from the clients) are sent to the leader. It then replicates the transactions on the rest of the network. Upon receiving a transaction, the leader insert it as a new entry to its log. The transaction is identified using the leader's current term and an index at which it is inserted to the log. Subsequently, the leader broadcasts the entry to all of the followers. Upon receiving the entry, the followers inserted it to their logs, and responds the leader with an acknowledgement receipt. The leader ensures that the entry has been replicated on a majority of nodes by counting the acknowledgement it receives. Once it receives one from $f+1$ or more nodes, it executes the comment contained in the entry. This also commits all other entries preceding the entry in question. The leader keeps track of the highest index it has committed, and includes such information in subsequent messages it communicates with the followers. This is to inform the later on the committed entries. Similar to [13], by running the Raft consensus protocol inside a TEE that offers attested and isolated execution, RVChain restricts adversarial behaviours of the faulty nodes, thereby reducing the threat model to crash fault tolerance, to which Raft applies [2, 26, 16].

3.3 Scaling Throughput with Parallel Shadow Chains

The RVChain comprises multiple parallel shadow chains. Network participants, or nodes or verifiers, in the RVChain are assigned to a particular shadow

chain uniformly at random using a random seed `rnd` generated inside their enclave [32, 25, 31, 30]. Given `rnd`, the nodes derive their shadow chain assignment by evaluating a random permutation π of $[1 : N]$ seeded by `rnd` (with N being the total number of network participants). π is then divided into approximately equally-sized chunks, each of which represents the verifiers-to-shadow chain assignment.

It is worth emphasizing that there is no upper limit for the number of shadow chains running in parallel. This effectively unleash the transaction throughput of the RVChain , allowing the processing capacity to grow in proportion with the number of verifiers joining the network.

Random Seed Generation. The RVChain exploits TEEs to obtain `rnd` in an efficient manner. The process is partaken by all nodes in the network, thereby assuring the fairness. The Random Seed Generation requires each node to be equipped with a RandomnessBeacon enclave, which is programmed to return fresh, unbiased random numbers subject to a certain probability. Similar to prior researches [17, 22, 35], this work considers a synchronous network (i.e., the communication delay Δ is known a priori) during the distributed randomness generation procedure.

To obtain its shadow chain assignment, each node in the network invokes its RandomnessBeacon enclave with an epoch number e representing the current period of time. Given the input e, the RandomnessBeacon enclave samples two random values `q` and `rnd` via two independent invocations of the `sgx_read_rand` function. Should `q` $= 0$, the enclave outputs a signed certificate containing $\langle e, \texttt{rnd} \rangle$. Otherwise, it returns $\perp$ signalling the node cannot obtain the certificate. Upon obtaining the certificate, a node broadcast it to the network. After a time Δ, all nodes in the network should have received all certificates that have been sent. They lock in the lowest `rnd` they receive for epoch e, and employs that value to evaluate its shadow chain assignment [14].

The security of this procedure is properly analysed in [14]. The RandomnessBeacon enclave is programmed in such a way that a node can only invoke it once every epoch. This prevents the adversary from selectively discarding the enclave's output so as to bias the final randomness. In an unlikely event wherein all nodes fail to receive any message after Δ (i.e., when no node can obtain $\langle e, \texttt{rnd} \rangle$ from its enclave), the nodes increment e and repeat the process. The probability of such an event is $P_{\text{repeat}} = (1 - 2^{-l})^N$ where l is the bit length of `q`. This probability can be configured in order to achieve a desirable trade-off between P_{repeat} and the communication overhead, which is $O(2^{-l}N^2)$. For instance, setting $l = \log(z)$ for some constant z, we obtain $P_{\text{repeat}} \approx 0$ and the communication is $O(N^2)$. Alternatively, if we set $l = \log(N)$, then $P_{\text{repeat}} \approx \mathrm{e}^{-1}$ and the communication is $O(N)$.

Securing individual shadow chain. Nodes on the same shadow chain engage in a consensus protocol to arrive at an agreement on the total order of the

transactions incurred on that chain. If a transaction is marked as "sensitive", it is processed inside the enclaves with isolated execution, thereby attaining confidentiality protection.

Global Order across parallel shadow chains We draw the neat idea of establishing one global order for all transactions posted on all shadow chains from [34]. In particular, transactions in each individual shadow chain are grouped into blocks. Each block is associated with two fields, namely (rank, NextRank), and a chain id, which is the id of the shadow chain it belongs to. These two fields are used to establish total order of transactions across the shadow chains. In the total ordering of fully-confirmed blocks, the blocks are ordered by increasing rank values, with tie-breaking based on the chain ids [34].

Nodes in the network are able to observe all the chains. Thus, they can infer the expected rank of the next block to be recorded on each shadow chain. Let us denote by x the largest value among such expected ranks, then x naturally associates with the "longest" shadow chain among all the shadow chains. The intuition is that every new block B should help its chain catch up with the current "longest" chain. Following this intuition, the node that proposes the new block B should set the NextRank field of B to x, or a value greater than x. It is also worth emphasizing that B's NextRank should always be larger than B's rank. This constrains is necessary to ensure rank values of blocks on each shadow chains are monotonically increasing.

Given the (rank, NextRank) fields of blocks are set in the afford mentioned manner, establishing a total order among blocks inhabiting different shadow chains is rather straightforward. For the sake of exposition, let us consider a local view of an honest node at any given time. We denote by y_i the value contained in the NextRank field of the last partially-confirmed block on the shadow chain i, and ConfirmBar be the minimum among all such values. It follows from the setting of (rank, NextRank) that next partially-confirmed block on any shadow chain must have its rank equal to or larger than ConfirmBar value. Consequently, one can consider all partially-confirmed blocks that have rank value smaller than ConfirmBar as fully-confirmed. These fully-confirmed blocks are ordered by their rank values. Should two blocks have equal rank values, tie is broken by their chain ids.

4 Incentives

Every blockchain needs a native token. For the sake of exposition, let us call native token of RVChain by RT[1].

The RT tokens shall be subject to the hard cap. A portion of the tokens are pre-minted for the development of the RVChain. The remaining portion of the token supply is reserved for rewarding network participants (or verifiers)

[1]The name of the token in deployment maybe different, and we will announce on the Ritva website once its name has been finalised

who contribute their resources to verify the transactions on the RVChain. It is worth mentioning that all transactions incurred on the RVChain network shall be subject to the transaction fee to be collected in RT. A portion of the transaction fee is given to the network participants, while the other is burnt off. This token burn inevitably leads to the depreciation of token supply over time, which translates into an appreciation of the token price against fiat.

References

[1] Ethereum: Blockchain app platform. `https://www.ethereum.org/`.

[2] The Coco Framework. `http://aka.ms/cocopaper`.

[3] Trusted computing group. `http://www.trustedcomputinggroup.org/`.

[4] Intel software guard extensions developer guide. `https://download.01.org/intel-sgx/linux-1.7/docs/Intel_SGX_Developer_Guide.pdf`, 2018.

[5] Tiago Alves and Don Felton. Trustzone: Integrated hardware and software security. Technical report, ARM, 2004.

[6] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for cpu based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, volume 13. ACM New York, NY, USA, 2013.

[7] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchain. `https://arxiv.org/abs/1711.03936`, 2018.

[8] Johannes Behl, Tobias Distler, and Rüdiger Kapitza. Hybrids on steroids: Sgx-based high performance bft. In *EuroSys*, 2017.

[9] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. *https://github.com/ethereum/wiki/wiki/White-Paper*, 2014.

[10] Tushar Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: and engineering perspective. In *PODC*, 2007.

[11] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *arXiv preprint arXiv:1804.05141*, 2018.

[12] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. `https://eprint.iacr.org/2015/564.pdf`.

[13] Hung Dang and Ee-Chien Chang. Autonomous membership service for enclave applications. *arXiv preprint arXiv:1905.06460*, 2019.

[14] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data*, pages 123–140, 2019.

[15] 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 *CCS*, 2016.

[16] Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 bft protocols. In *EuroSys*, 2010.

[17] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger. *IACR Cryptology ePrint Archive*, 2017.

[18] Jae Kwon. Tendermint: Consensus without mining. `https://tendermint.com/static/docs/tendermint.pdf`.

[19] Leslie Lamport. Fast paxos. *Distributed Computing*, 2006.

[20] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[21] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.

[22] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *CCS*, 2016.

[23] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. *HASP@ ISCA*, 10, 2013.

[24] Silvio Micali. Algorand: the efficient and democratic ledger. *arXiv preprint arXiv:1607.01341*, 2016.

[25] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *FOCS*, 1999.

[26] JP Morgan. Quorum. `https://github.com/jpmorganchase/quorum`.

[27] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[28] Diego Ongaro and John K Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, pages 305–319, 2014.

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

[30] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *CRYPTO*, 1999.

[31] Markus Stadler. Publicly verifiable secret sharing. In *EUROCRYPT*, 1996.

[32] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *IEEE S& P*, 2017.

[33] Florian Tramer, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In *EuroS&P*, 2017.

[34] Haifeng Yu, Ivica Nikolic, Ruomu Hou, and Prateek Saxena. Ohie: blockchain scaling made simple. *arXiv preprint arXiv:1811.12628*, 2018.

[35] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *CCS*, 2018.