On or Off the Blockchain? Insights on Off-Chaining Computation and Data

Jacob Eberhardt and Stefan Tai

Information Systems Engineering (ISE)
TU Berlin, Germany
{je,st}@ise.tu-berlin.de

Abstract. The potential for blockchains to fundamentally transform how organizations produce and capture value is huge and very real. Practical applications dealing with nearly any type of digital asset demonstrate this capacity. Blockchain-based application architectures benefit from a set of unique properties including immutability and transparency of cryptographically-secured and peer-recorded transactions, which have been agreed upon by network consensus. Blockchain-based applications, however, may also suffer from high computational and storage expenses, negatively impacting overall performance and scalability. In this paper, we report on lessons learned and insights gained from a set of experimental blockchain projects, focusing on off-chaining: How to move computation and data off-the-chain, without compromising the properties introduced and benefits gained by using blockchains in the first place.

1 Introduction

Blockchains are a combination of different computing and economics concepts, predominantly including peer-to-peer networks, asymmetric cryptography, consensus protocols, decentralized storage, decentralized computing and smart contracts, and incentive mechanisms. The synthesis of these concepts positions blockchains as a new technology and as a programmable platform and network at the same time. Blockchains introduce unique properties including immutability and transparency of cryptographically-secured and peer-recorded transactions, which have been agreed upon by network consensus. As such, the potential associated with blockchain to fundamentally transform how organizations produce and capture value is huge – and very real. While initially discussed especially in the financial services sector, there are practical applications today dealing with nearly any type of digital asset, ranging from asset provenance to peer-to-peer commerce. Establishing trustless interactions and business disintermediation remain major objectives when using blockchains.

Over the past two years, we have conducted a set of blockchain projects at the Information Systems Engineering research group at TU Berlin, mostly experimental projects that focus on a particular application challenge and which have been carried out in collaboration with industry partners. Further, we are conducting foundational research on blockchain technology and platforms, taking

a distributed systems and data management perspective. In both settings, the question of on-chaining versus off-chaining is recurring: What exactly has to be on the chain and what can be off the chain, while retaining the overall properties and benefits associated with blockchains?

On-chain data – in the form of confirmed transactions organized in ordered blocks – and on-chain code – in the form of programs written in a general-purpose, Turing-complete language – require validation and consensus by network peers and result in append-only changes to the blockchain as a shared datastore that cannot be reversed. Transaction validation, consensus protocols, and decentralized program execution may, however, describe a communication and execution overhead. And they simply do take time. In addition, miners (that is, nodes that validate transactions and propose new blocks) typically charge fees, thereby incurring financial costs. Overall, scalability of the blockchain-based system may suffer. Bitcoin currently has a limit of 7 transactions per second; Ethereum has a limit of about 15 transactions per second. Furthermore, anything on the (public) blockchain is not inherently anonymous, but on the contrary purposely visible; privacy and confidentiality are not guaranteed for on-chain transactions.

The objective for off-chaining data and computation is to reduce or to overcome such limitations. By moving data and computation elsewhere off the blockchain, for example, to another datastore, server, or third party, the blockchain "footprint" obviously is reduced. However, the fundamental properties of blockchains and blockchain-based applications, may be compromised to different degrees when doing so. They may even be potentially prohibitively violated when using naive off-chaining approaches. After all, the system should remain "trustless" in the sense that no explicit trust is required.

In this paper, we report on first insights gained on off-chaining computation and data. We present five off-chaining patterns and discuss the context, principle idea, and implementation for each.

2 Blockchains and Smart Contracts

In a nutshell, blockchains are distributed peer-to-peer systems which implement a trustless shared public append-only transaction ledger [14].

Blockchains. Bitcoin, the first implementation of such a system, was proposed in 2008 by Satoshi Nakamoto [10]. The goal which led to the creation of the Bitcoin protocol was the design of a digital currency which allows the transfer of digital value fully peer-to-peer without relying on a trusted intermediary. To implement such a decentralized cryptocurrency, the system combines transactions secured by asymmetric cryptography with a consensus algorithm to decide on the transaction order within the network. Peers in the network can validate individual transactions by checking cryptographic signatures. In addition to that, however, a global order of the transactions has to be decided on to prevent double spending of digital funds. For that, as a main innovation of the Bitcoin system,

the proof-of-work consensus protocol was developed which allocates the right to add transactions to the network in proportion to the computational effort spent to secure the network. For efficiency, multiple transactions are grouped into a block. These blocks are then ordered by consensus. Each block references its predecessor, which implies a chain data structure – the blockchain.

Extending Bitcoins idea of peer-to-peer value transfer, Ethereum, a trustless computing platform, was proposed in 2014 [9, 16]. Ethereum adds a turing complete and stateful programming language to the blockchain idea, which enables the execution of complex code without trusting a server or central party. Trust is replaced by validating each program execution on every peer in the network and agreeing on an outcome.

Smart Contracts. These programs executed in a trustless and tamper-proof manner in the network are referred to as Smart Contracts. Note, that Smart Contracts need to be deterministic as otherwise peers could disagree on the results of valid executions. Hence, e.g., filesystem and network access are not permitted. While the term may imply a close connection to legal contracts, smart contracts have a much wider range of use cases and can be applied where automatically executed complex conditional logic is required. Hence, they can be imagined as self executing autonomous agents.

3 The Need for Off-Chaining Computation and Data

Over the last two years, we gained extensive experiences during proof-of-concept implementations of blockchain-based applications prototypes. We motivate off-chaining insights by discussing exemplary challenges for two of these applications. While all prototypes developed at TU Berlin have been realized on the Ethereum platform [9], we consider our findings to be representative for all of today's public blockchain implementations.

As a first application, we created a fair and manipulation-resistant chess game on the Ethereum platform [1].

Today, online gaming relies on a trusted intermediary which runs games and ensures players obey the rules. However, this intermediary needs to be trusted to not cheat or steal funds. To eliminate that trust, we implemented the chess logic as well as data structures required to persist the game state in a smart contract. Conceptually, that already solves the problem: Players send moves to the contract, which modifies game state persisted internally for valid moves. After a valid move, the contract checks end game conditions and pays out the winner if a condition is met. Checking end game conditions, however, is computationally expensive. All potential moves have to be calculated and verified to check the check mate condition. This is not possible in a smart contract as it violates the complexity upper bound for on-chain transactions.

Three things can be learned from that:

1. We need to find a way to perform the end-game check off chain, on the client side, without impacting the blockchain's trustlessness property.

2. As computations come with a fee, the end game check should be performed as rarely as possible. Hence, like in a physical chess game, we should have a player trigger the check instead of doing it after every valid move. In other words, we should move part of the control flow to the client side.
3. We have to expect to reach scalability limits of blockchains when creating applications. This emphasizes the need for research in and development of off-chaining techniques.

Another much more complex proof-of-concept was a decentralized service marketplace which enables trustless disintermediation between providers and consumers of service APIs. Using a cryptocurrency for payments, a consumer can buy time-constrained access to a service offered on the market place without involving a marketplace intermediary. Especially service discovery, one of the main building blocks of a service marketplace, posed a big challenge within the fully decentralized design: Data storage on blockchains is extremely expensive due to full replication in the peer-to-peer network. Nonetheless, a meaningful service discovery feature requires API descriptions to be stored. Simply pointing to an off-chain reference from a smart contract, e.g., a file hosted in a cloud storage system, is no alternative to on-chain storage. This approach would introduce trust in the storage system since the data stored could change while the reference remains the same. Additionally, since all data in a blockchain is stored on every node in the network, it is publicly visible. There is no obvious way for a service provider to hide some of his service descriptions from the public.

As a direct consequence of this public visibility, there is no way to perform computations on private data on-chain without revealing it. Assume a consumer wants to prove to a provider that he has access to another provider's API. That second provider could publicly provide the hashes of all tokes that give access to his service. Then, the consumer could simply hash his private access token and show the hash to the first provider, which could in turn verify it by comparing it to the published hashes. However, for this to be trustless, the consumer would need to perform the hash operation on-chain and with that reveal his private token. Simply computing the hash off-chain would not proof anything to the provider.

Again, we derive three challenges from these findings:

1. We need to find a way to store data off the chain without giving up its manipulation-resistance.
2. As all on-chain data is publicly visible, techniques for trustless but privacy-preserving off-chain storage should be developed.
3. Off-chain computations on private data which can be verified on-chain without revealing said data would augment the set of possible use cases.

In summary, off-chaining strategies are needed to address both, functional limitations of and high costs incurring from on-chain computation and storage.

4 Off-Chaining Patterns

We now introduce a set of off-chaining patterns identified, which can be used individually or in combination to move computation and data off the blockchain. Each pattern aims at maintaining the key properties of blockchains and includes techniques to ensure that they are not compromised to an unwanted degree.

4.1 Challenge Response Pattern

Context: A smart contract models a state machine with well-defined final states. State transitions are cheap to compute, but checking whether a given state is a final state is expensive.

Solution: Instead of checking whether a state is final or not in a smart contract on a blockchain, the same check is performed off-chain on the client side. A client can notify a smart contract when a final state has been reached. Other clients can prove claims wrong by providing a valid state transition. Using this pattern, the computation never has to be performed on-chain.

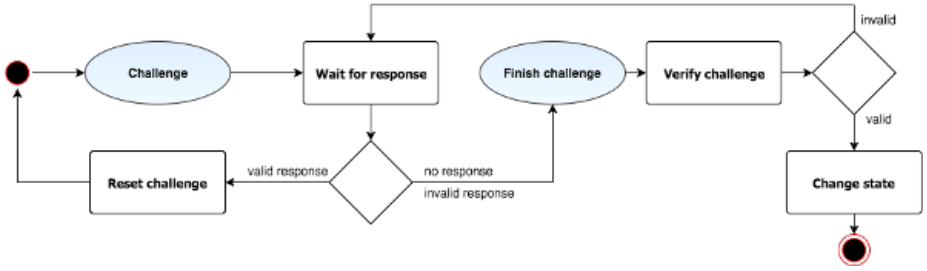

Fig. 1. Challenge Reponse Pattern

Example: The end game condition for chess is too expensive to check on-chain. The players, however, can easily check the condition off-chain. Hence, instead of checking the end game condition in a smart contract, a player simply claims check mate. If the claim was false, his opponent can simply prove him wrong by submitting a valid move. If the claim was true and the opponent cannot submit a valid move, the winner is paid out. Figure 2 gives an overview of the full challenge response protocol for chess also considering draws and timeouts. For a more detailed description, we refer to [1, 2].

Discussion: This pattern allows computations to be off-chained efficiently in scenarios where smart contracts act as state-machines. Since it allows for complex operations to be moved completely off-chain and with that circumvents

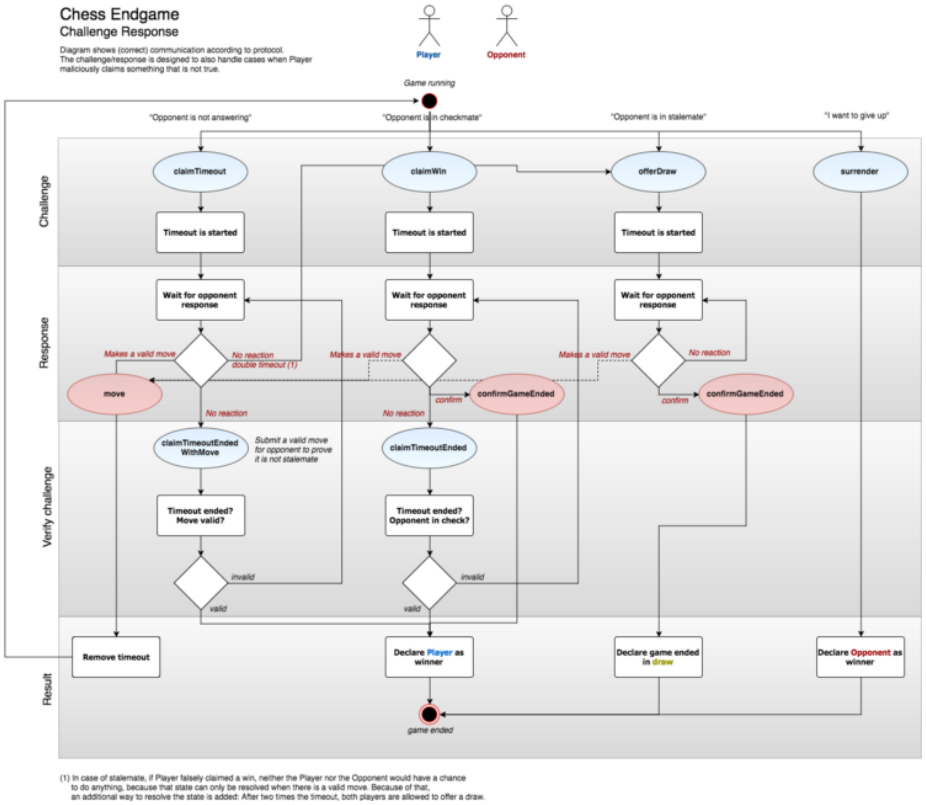

Fig. 2. Challenge Response Pattern applied for Chess

the complexity upper-bound for on-chain transactions, it can extend possible use cases and potentially lead to cost savings. Note though, that the pattern increases the overall amount of on-chain transactions, which requires a careful calculation of costs. Also, increased availability of the parties involved in the smart contract implementing the pattern is required, since the use of timeouts is essential to ensure progress.

Implementation: This pattern does not require additional technologies besides smart contracts. For an exemplary implementation of this pattern, refer to [2].

4.2 Off-chain Signatures Pattern

Context: Two network participants know that they will perform a set of transactions in the future. They want to reduce the cost of these transactions or want to hide them from other network participants.

Solution: Together, the two participants specify a smart contract including a function, which applies an external state given as argument to the contract state. This function includes a signature check to ensure both participants agree with the state change. Only if valid signatures of both participants are supplied with a requested new state, the new state is applied. This contract is deployed to the blockchain and both participants optionally make a deposit.

Then, the participants perform transactions purely off-chain and peer-to-peer, without involving the blockchain: One participant computes a new state, wraps it in a transaction, signs it and sends it to his counterpart. The recipient then checks the new state, signs the transaction as well in case he agrees and sends it back to the sender.

This transaction, signed by both parties, can now be sent to the smart contract by a participant at any point in time. After validating both signatures, the contract updates its state accordingly.

Example: Participants A and B create a smart contract with a signature-locked state update function and deposit 50 units of cryptocurrency each. Now, A wants to transfer 10 units to B. For that, she creates a transaction locally, which includes a new state where A and B have balances of 40 and 60. She signs it and sends it to B, who signs it as well. Now, B can use the transaction to update the on-chain balance at any point in time. However, A and B could perform further off-chain value transfers without ever settling on-chain unless one side's deposit is used up. This application of the pattern for off-chain value transfer is often referred to as payment channel.

Discussion: This pattern allows efficient off-chain transactions without introducing trust into the system. The core insight is, that the guarantee to be able to settle a transaction is as good as actually executing the transaction on-chain. Signing a new state is analogous to writing a check in a traditional financial transaction. Using off-chain transactions can lead to significant cost savings as transaction fees only apply for on-chain settlement. Furthermore, the pattern can enhance privacy and confidentiality, as all transactions but the final settlement remain hidden from the network. From a blockchain network perspective, this pattern helps to take load off the system and with that enhances scalability.

There are many other applications besides simple value transfers. As shown in figure 3, we were able to move the core parts of the chess game off-chain by using this pattern. This not only helped to lower the cost of a game, but also to remove time dependence on block intervals.

Since initial deposits to the smart contracts are required in most cases, establishing contracts with many peers can lock a considerable amount of funds. Also, malicious participants could freeze funds by denying signatures. Hence, contracts should specify timeouts which trigger automatic settlement.

Implementation: Besides on-chain smart contracts, this pattern requires a peer-to-peer communication channel to exchange signed off-chain transactions.

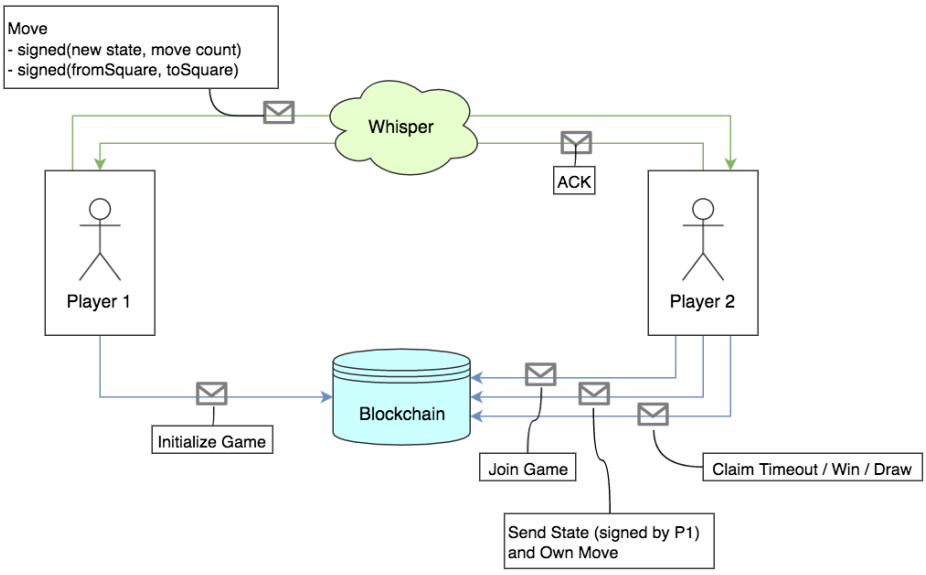

Fig. 3. Off- and on-chain interactions in the chess application

In the Ethereum ecosystem, for example, the Whisper Messaging Protocol [4] can be used. There are various efforts to leverage this pattern to build off-chain value transfer networks for existing blockchains: The lightning network [12] provides an implementation for the Bitcoin ecosystem, while Raiden [5] targets the Ethereum network.

4.3 Content-Addressable Storage Pattern

Context: A large amount of data is associated with a smart contract. On-chain storage is too expensive.

Solution: Store the data off-chain in a content-addressable storage system and store the reference in the smart contract. Clients using the smart contract can retrieve the reference and based on that retrieve the data. Then, they can verify the data's correctness by recomputing its address from itself and comparing it to the reference stored in the smart contract.

Example: A smart contract encodes ownership of a piece of digital art. However, a piece of art would be very expensive to store on-chain due to its size. To solve this problem, the description is stored in a content-addressable storage system which stores files by their hashes. The file hash is also stored in the smart contract, serving as a reference to the artwork. Clients can then retrieve the hash of the externally stored piece of art from the contract and use it to

query the storage system. The result can then simply be hashed to verify its correctness.

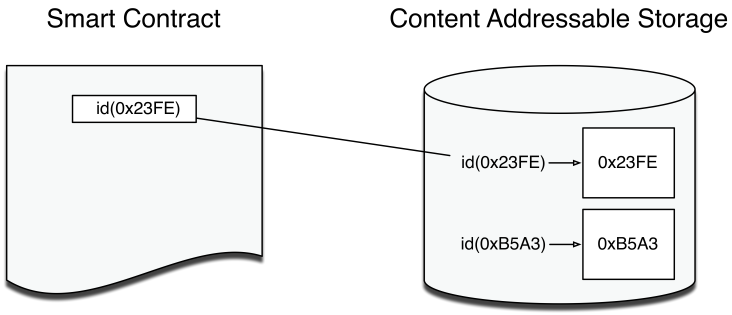

Fig. 4. Content-Addressable Storage Pattern

Discussion: This pattern allows the trustless outsourcing of data to an off-chain storage system since a modification in the data would immediately change its address and with that invalidate its references.

By applying the pattern, an application's storage cost can be greatly reduced and files, which originally could not be stored on-chain in the first place, can now be referenced without introducing trust. Additionally, as the data retrieval is done on the client side from an external storage system, privacy features may be implemented by adding access control to that system. However, this requires careful considerations depending on the use case, since leaked data can immediately be confirmed to be authentic by recalculating its address.

While not in scope of this pattern, the required external content-addressable storage system itself has to be reliable and available. In case of unavailability or data-loss, the blockchain-based part of the application may also become unavailable.

In the future, this pattern could be extended to support trustless computation on data stored off-chain: First, content-addressed data referenced from a smart contract could be sent to the contract. Then, integrity could be verified on chain. In case of success, the smart contract could modify the data, update its reference to that new data and write it to an event. An untrusted external worker could then write that data back to the content-addressable storage system the inputs were retrieved from. While theoretically interesting, we did not yet observe this extension. Hence, it is not part of the pattern.

Implementation: As mentioned before, a content-addressable storage system is required to work in conjunction with smart contracts. Two such technologies, which address data by its hash and try to ensure availability and durability are the Interplanetary File System (IPFS) [7] and Swarm [15].

4.4 Delegated Computation Pattern

Context:

a) A node participating in a blockchain network wants to prove a property of its private data without publishing it.
b) A node wants to perform a computation that is too complex to be executed on-chain.

Solution: Outsource computation to an untrusted third party and, besides the result, generate a proof of correct execution. Instead of executing the computation itself, verify the proof of correct execution on-chain.

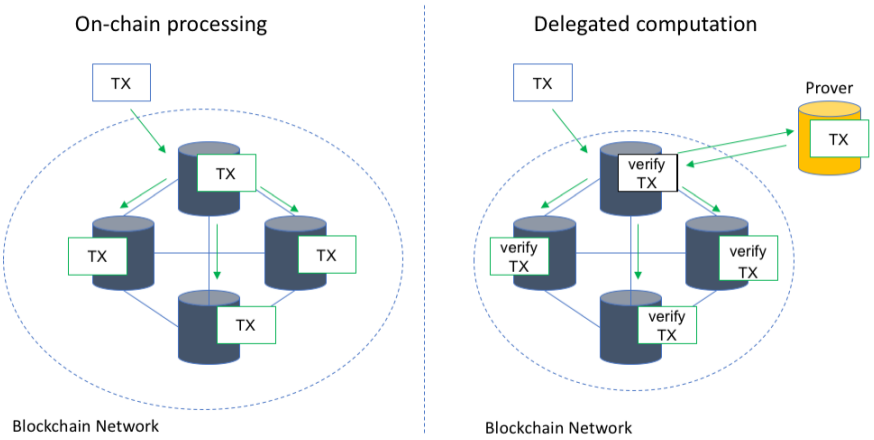

Fig. 5. On-chain processing vs. Delegated Computation Pattern

Example: There is an on-chain list of hashes of ID-card information which refers to people who are allowed to call a smart contract function. Now, anyone listed can prove that he has an ID-card which authorizes him to call the contract function by hashing his card information locally and supplying the result including a proof of correctness. The proof does not require to reveal any of the information on the card.

Discussion: This pattern allows the trustless outsourcing of computation to untrusted parties. The third party, also called the prover, does not have to reveal any private inputs or intermediate results of the proof creation. The only information leaked is that the prover knows all the information necessary to

correctly compute the output. For that, non-interactive zero-knowledge proofs, more specifically zero-knowledge Succinct Non-interactive ARgument of Knowledge (zkSNARKs), can be employed [11, 8].

Unlike with regular computations on the blockchain, this pattern allows off-chained computations to hide information used during execution. Hence, not having to expose information but the result of a computation greatly enhances privacy. Furthermore, the proofs can be designed in a way that the verification cost is independent of the complexity of the off-chained computation. Thus, after a complexity threshold is reached, on-chain verification of a computation is cheaper than its on-chain execution. This result can be leveraged to increase a blockchain's throughput. Even operations exceeding the on-chain complexity limits for computations may still be executed off-chain using this patterns.

The state of the art non-interactive zero-knowledge proofs require a trusted setup phase to be performed before proofs can be generated. This can, depending on the use case, introduce undesirable trust in the overall system. Additionally, the proof generation for a computation causes an overhead over its non-verifiable execution. Yet, there is neither a high level language for the convenient specification of off-chain computations nor are there tools for simple on-chain verification of proofs. Therefore, while powerful and already used in practice, this pattern is currently only applied in rather specific scenarios, e.g., in zCash, which is a Bitcoin-based blockchain which implements privacy preserving transactions [6, 13].

Implementation: For the verification of off-chain computations from smart contracts, the underlying blockchain needs to support the operations needed to check proofs. These can either be use case specific, or universal building blocks which can be used to verify any proof. While zCash directly added the verification logic for their specific computation to their protocol, Ethereum plans to add operations to support verification of arbitrary zkSNARKs with the Ethereum Improvement Proposals 196 and 197 [3].

4.5 Low Contract Footprint Pattern

Context: Changing a smart contract's state requires an on-chain transaction. To incentivize the processing of a transaction by the network, a fee has to be paid. This fee depends on the complexity of the smart contract function called as well as its use of storage.

Solution: To optimize fees, contracts should be designed in a way that minimizes the number and size of on-chain transactions. The following two techniques can be used to reduce the footprint.

- Do not check conditions on-chain after a state change. Let nodes perform the condition check locally and trigger an on-chain check in case of success.

– Optimize for writes, not reads. Reading from a smart contracts is a local off-chain operation and does not require an on-chain transaction. Minimize writes and store information free of redundancy. Compute derived data locally during reads.

Examples:

– In the service marketplace application, a service provider needs to make sure consumers are removed from the on-chain authorization list after the time period the consumer paid for is over. Instead of periodically triggering or linking the condition check to another contract function and risking frequent reevaluation, he tracks the access period locally and triggers the on-chain check after it has elapsed. This reduces the amount of on-chain evaluations to one.
– If the service provider wants to know the number of customers currently subscribed to his service, he should not add a counter to the smart contract. He can compute the number locally at any point from the authorization list. This saves storage space and counter update operations.

Discussion: This pattern may not initially seem like an off-chaining approach, as it does not explicitly take something off the chain. However, it prevents information to be stored or processed on-chain in the first place. Hence, this may be the least obvious, but the most employed and intuitive off-chaining pattern.

Implementation: No additional components or techniques are required besides smart contracts to implement this pattern.

5 Summary and Outlook

In this paper we motivated the need for off-chain approaches to overcome limitations in today's blockchain implementations and even more, to extend their functionality and to reduce usage costs. After deriving key challenges based on our experiences from implementing several blockchain-based applications, we presented five off-chaining patterns for moving computation and data off the blockchain, without compromising important blockchain properties, in particular, the trustlessness property.

We expect blockchain systems to further mature and improve with regards to scalability and privacy in the future by combining and implementing ideas like, for example, new consensus algorithms, sharding, or homomorphic encryption. However, we still consider off-chaining techniques to be key tools in blockchain-based application engineering as they introduce additional functionality and potentially significant cost benefits.

References

1. Chess on Ethereum. https://medium.com/@graycoding/lessons-learned-from-making-a-chess-game-for-ethereum-6917c01178b6, accessed: 2017-06-26
2. Ethereum Chess Proof-of-Concept Implementation. https://github.com/ise-ethereum/on-chain-chess, accessed: 2017-06-26
3. Ethereum Improvement Proposals (EIPs). https://github.com/ethereum/EIPs, accessed: 2017-06-22
4. Ethereum Whisper Protocol v5. https://github.com/ethereum/go-ethereum/wiki/Whisper, accessed: 2017-06-27
5. Raiden Network. http://raiden.network/, accessed: 2017-05-12
6. zCash. https://z.cash/, accessed: 2017-06-20
7. Benet, J.: IPFS - content addressed, versioned, P2P file system. CoRR abs/1407.3561 (2014), http://arxiv.org/abs/1407.3561
8. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 326–349. ITCS '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2090236.2090263
9. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper (2014)
10. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
11. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 238–252. IEEE (2013)
12. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network (2015), accessed: 2017-05-12
13. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: Security and Privacy (SP), 2014 IEEE Symposium on. pp. 459–474. IEEE (2014)
14. Tai, S., Eberhardt, J., Klems, M.: Not ACID, not BASE, but SALT - a transaction processing perspective on blockchains. In: Proceedings of the 7th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,. pp. 755–764. INSTICC, ScitePress (2017)
15. Trón, V., Fischer, A., Nagy, D.A., Felföldi, Z., Johnson, N.: Swap, swear and swindle - incentive system for swarm (2016)
16. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper (2014)