Off-Chain Smart Contracts

Mohanraj Polur Balu and Umashankar Somasekar

[1]RapidQube Digital Solutions Pvt. Ltd., Mumbai, India

mohanraj.polurbalu@rapidqube.com
umashankar.somasekar@rapidqube.com

Abstract. Smart Contracts should surpass their limitations and evolve to changing architectural standards that are needed for blockchains to scale significantly while retaining their decentralized principles. The computing complexities imposed on smart contracts such as decision making through predictive analytics, machine learning and dependencies on Oracles, could extend the transaction throughputs and potentially limit the scope and usage of smart contracts across various business processes. Our intent through this paper is to present one of the ideas termed "Off-Chain" contracts that could isolate the computing complexities from the blockchain by keeping the code and execution of the contracts away from the chain, but their hashes and log On-Chain thereby preserving the fundamental principles of decentralized ledger technology..

Keywords: Contract, Smart Contracts, Blockchain, Off-Chain, On-Chain, Synthesized assets, Intelligent assets, Wet Code, Dry Code

1 Introduction to Smart Contracts

1.1 Preamble

Immutable records of distributed consensus, smart contracts, and intelligent assets are becoming the new order of the world. The need for special focus on Data Governance, Integrity and Integration may become a thing of the past. Smart contracts, the self-executing, self-enforcing, Computer Protocols will define these rules and scrutinize the adherence of transactions over the assets using the widely accepted principles. Smart Contracts have the ability to minimize arbitrarily executed rules that could derail trust, the core element of any contract. Decentralized and immutable contracts could provide better opportunities for the world communities to create and sustain a happier and safer community.

1.2 History

The phrase "smart contracts" was coined by Nick Szabo in 1996. “The basic idea of smart contracts is that many kinds of contractual clauses (such as liens, bonding, delineation of property rights, etc.) can be embedded in the hardware and software we

deal with, in such a way as to make breach of contract expensive (if desired, sometimes prohibitively so) for the breacher."

Also, smart contracts, as opposed to paper-based legal contracts, automatically perform the obligations the parties have committed to under their agreement. The blockchain is the ideal place to store such a contract, both because of its immutability and its cryptographic security. This store guarantees high assurance and reliability of the digital and self-executing contract protected from undue changes and failures.

1.3 Smart Contract in action

Here's an example to consider. I am renting a car for a day. The Vendor issues a smart contract. Both of us agree to oblige by the clauses and parameters defined in it by digitally signing the contract. This smart contract may hold many critical functions such as–

1. Establish the identity of car, renter, and vendor
2. Transfer the digital key from vendor to the renter to unlock the car.
3. Create an escrow account between the renter and vendor.
4. Test the clauses in the smart contract against the parametric data fed by the Electronic Control Units fitted in the car and report any violation.
5. Auto-debit the escrow account towards the rental fee.
6. Auto-debit the escrow account based on any contractual clause violations.
7. Transfer the digital key from renter to the vendor to lock the car.
8. Issue a closure of the contract.

Once the contract is initialized, the involved parties can be assured that this agreement will be programmatically enforced. Storing this Smart Contract on a blockchain ensures that it cannot be influenced by either of the parties. Thus, the digital alternative of a notarized paper-based Contract proves worthwhile to be implemented.

Smart contracts are poised to revolutionize many industries by replacing the need for both traditional legal agreements and centrally automated digital agreements..

2 Background

One cannot deny the fact that Smart contracts are as only as good as they are programmed.

The smart contract enabled blockchain responds efficiently as a "trusted third party" while providing transaction security, but it could lack consistency, performance and might be susceptible to hacks if not designed properly.

2.1 Exploits

The cryptocurrency world has witnessed some of such attacks; few of them are listed below:–

Attack	Period	Exploit/ Cause	Loss/Impact
Bitcoin Attack	August 2010	Bug in Bitcoins software	• Created a single block involving a transaction of 184 billion Bitcoin. • Bitcoin network forked to an updated version of the bitcoin protocol
DAO Hack	June 2016	Vulnerability in the DAO code	• Theft of $50 million in Ether • Hard fork in Ethereum into Ethereum and Ethereum Classic
Tangerine Whistle / Spurious Dragon	October 2016	DoS attack by repeatedly calling certain operation codes in smart contracts	• Network filled with pending transactions causing delays in processing transactions • size of the blockchain inflated • Hard fork of Ethereum

2.2 Performance

From a performance perspective, smart contracts can be programmed only to handle computational problems of less complexity. Programming smart contracts to handle analytics, modeling, intelligence, etc., could result in poor transaction performance and lead to a lack of producing results in real-time.

For example, most online games work using a third-party to establish communications and compatibility. They make sure everything works correctly. This is third-party that players have to trust to do the right thing. A public blockchain can eliminate the third-party, and record all the gaming transactions and events on a global shared and distributed ledger.

A complete game of Chess can be played reliably On-Chain, meaning no other communication is involved except messages to and from the Smart Contract. For checkmate (the king is in check and player cannot make any move to escape) and stale-mate (same, except that the king is not in check), to verify the condition all possible moves have to be calculated and verified, which is unfeasible on the

blockchain. When implemented on an On-Chain Smart contract, the logic resulted in transaction time exceeding the current limits set for the network.

Computationally "expensive" calculations are usually not suitable to be run On-Chain.

2.3 Real-time processing

The transaction rates of blockchain aren't yet addressing the real-time needs of business problems. Bitcoin handles 3-4 transactions per second, and Ethereum handles 20-30 transactions per second. For a gaming world, this wouldn't be sufficient enough to feel real time.

The biggest problem in making completely On-Chain games, however, is the lack of real-time computational ability.

2.4 Oracles

The data providers of the events that should trigger smart contract cannot be on a centralized network which could derail the fundamental principles on which the technology has been built

3 Off-Chain Smart Contract

Off-Chain contracts are smart contracts whose code is not run by the miners, requesters, endorsers or harvesters, but rather the client. Interoperability consensus facilitates this process to-

- run data-intensive computations more cost-effectively
- run computations with high computational time without clogging the network
- write code that is not bound to a cryptocurrency incentive structure
- write code that has less impact on the blockchain.

We still want to keep our serverless architecture, so we can use peer-to-peer web protocols to achieve this. Let's think about solving the game problem.

As long as the game runs smoothly, i.e., each player is alternatingly sending a correct move; the blockchain need not verify all these correct moves-the clients can do that. All moves directly between the clients are sent to the blockchain. Each player should have the possibility to prove the current game state independently. To prohibit cheating, each report to the blockchain should consist of the complete game state with a valid signature from the other player. If the game is managed in this way it can be cryptographically proven that both players accept the current state.

In other words, during normal gameplay, each player keeps sending their next move and the resulting game state, plus a cryptographic signature. If the other player

disagrees, they can submit this data to the Smart Contract for verification. The same can be done at the end of the game. Furthermore, clients will acknowledge each other's messages to discover problems faster

Apart from setting up the game (Initialize and Join), clients only send messages to the Blockchain when they want to claim something (timeout, win, draw), and to claim their prize.

The application is still serverless, secure and trustless. In the end, the blockchain is the single source of truth, but the clients help in making it more efficient. In case of a problem, however, the game can fully degrade to run on the Blockchain.

The smart contract to verify the state can be stored and computed Off-Chain. Creating an identity for the Off-Chain smart contract on blockchain using code signing techniques secures it from being tampered as well as preserves the immutability property, the only reasons for which it is preferred to be On-Chain. Moreover applying principles of "proof-carrying code," the smart contract is verified for its identity just before it executes and modifies the state of the blockchain.

The identity of the code includes its metadata such as the hash of the code, version, and timestamp. These Off-Chain smart contracts also follows the lifecycle of an asset on a blockchain and enables the transfer of contract between the participants.

The above approach is analogous to a shared user library call by the kernel.

4 Off-Chain & Other Features of NEM

NEM is without a custom smart contract by design. This blockchain product believes it's a bad idea to execute code On-Chain for security and scalability reasons. This platform supports a predefined On-Chain smart contract that is as reliable as Multi-sign, and Off-Chain smart contracts.

NEM's design is lighter-weight and as such is faster. It strives to only put data and assets on the blockchain that makes sense. NEM comes with data management apps. built right into the client. These are called "smart assets". Smart assets allow you to create data records and tokens and voting systems with only a few clicks

NEM's approach is to let developers use a wide range of configurable functions which allows them to build powerful applications based on a closed set of atomic operations, and opens the network to almost any technological combination thanks to its REST API. Notably, mix matching and combining Namespaces (unique domains), Mosaics (customizable assets), 2.0 multisign contracts, and three forms of messaging. This allows for a wide variety of application frameworks to be built. By assigning meaning to these different functions and combining these in various ways a wide

range of applications are currently under development including apps. for transmitting financial value, notarizations, tracking and logistics, voting, land management, ID management, and more

5 References & Links

1. Ethereum VS NEM. The difference between Ethereum and NEM - https://steemit.com/ethereum/@steemthenews/ethereum-vs-nem-the-difference-between-ethereum-and-nem
2. Ethereum Versus NEM - The Obvious Choice https://blog.nem.io/ethereum-versus-nem-the-obvious-choice/
3. A survey of attacks on Ethereum smart contracts https://eprint.iacr.org/2016/1007.pdf
4. *Lessons learned from making a Chess game for Ethereum* https://medium.com/@graycoding/lessons-learned-from-making-a-chess-game-for-ethereum-6917c01178b6
5. Enabling Off-Chain AI for Ethereum https://medium.com/iex-ec/enabling-Off-Chain-artificial-intelligence-for-ethereum-with-iexec-804e640667c0