A technical description of the Smilo Platform

Elkan Roelen & Thomas Modeneis

Smilo Foundation, Rotterdam, The Netherlands

January 1, 2019

Abstract

Smilo is a unique full-featured hybrid blockchain platform that will be able to facilitate hybrid transactions, hybrid smart contracts and hybrid decentralised applications — with 'hybrid' referring to both public and private. Smilo uses their unique blockchain technology to facilitate an alternative protocol for decentralised applications, including GDPR compliant private vaults, infinite scaling and accurate cost predictions.

Keywords: Smilo, Blockchain, Private, Public, Decentralised, Private Smart Contracts, GDPR

Introduction

Ethereum was initially proposed in 2013 in a white paper[1] by Vitalik Buterin, a cryptocurrency researcher and programmer. In the white paper, Vitalik Buterin described Ethereum as a public, open-source, blockchain-based computing platform featuring smart contract functionality. Since the release of Ethereum in 2015, several other projects featuring smart contract functionality have emerged, but these platforms are unable to host both anonymous and public smart contracts.

For mass adoption of blockchain technology, the Smilo team firmly believes that there must be a connection between a blockchain and its actual use cases. In order to ensure this connection, it is important to choose the right platform to connect with these use cases. Currently, there is no suitable blockchain-based computing platform for the medical sector, nor one that is an all-in-one solution for private escrow arrangements. All of the current smart blockchain-based computing platforms feature public smart contracts, but few people prefer their medical records or escrow arrangements to be public.

To address these shortcomings, what is needed is an open-source, hybrid, blockchain-based computing platform: one that features hybrid transactions, hybrid smart contracts, and hybrid decentralised applications. This is where the Smilo Platform shines.

Smilo is based on the unique and self-made Smilo BFT+ consensus protocol. This consensus algorithm combined with the exclusive Smilo Network Protocol (SNP) ensures secure, scalable, quick, and sustainable transactions. Additionally, all the transactions will be free of charge enabling corporations to deploy engaging decentralised applications.

Besides, this consensus mechanism also empowers the community with the autonomy to run their own masternodes. Smilo can achieve complete decentralisation with enough independent masternodes, and through this network of networks the transaction processing capabilities of the platform will be over 3000 transactions per second, increasing with every additional masternode.

Key features

Free transactions

Smilo improved the Ethereum Gas mechanism after a thorough examination. With the implementation of these improvements a few new features quickly presented themselves, one of which are free of charge transactions. The mechanism works as follows: each account has the right to use the network resources of the platform. If an account decides to broadcast a transaction, the mechanism governs the speed of the transaction response to repel any inconsistencies. Thus, the transaction will be confirmed safely and without any hesitation. These transactions do not consume the basic currency of the network. Instead, they use SmiloPay which users will quickly stake.

GDPR compliant private Vault (private smart contracts)
The private smart contracts designed by the Smilo Platform facilitate GDPR compliances through side-chains between two or more parties. The execution and state of the contract will only be accessible by the owner and their permitted parties. By using a private side-chain, the state will not be saved on the main chain, although the state changes will be saved on the main chain. Thus, creating a secure environment for private smart contracts on side chains.

Open, decentralised and infinitely scalable
The Smilo network is based on masternodes accommodated by the Smilo Network Protocol. Every new node will automatically be accepted into the network to help protect the network. With the use of this protocol there will not be any limitations in regard to the number of masternodes.

Fast confirmations (instant)
The Smilo network uses the Smilo BFT+ algorithm combined with masternodes. Due to this advanced network topology, 99 percent of the transactions will receive confirmation within one single second, and the final transaction confirmation will only take up to three seconds! This confirmation speed is unknown when compared to other available blockchains.

High throughput
Based on the Smilo BFT+ consensus mechanism in combination with other specific blockchain related parameters, Smilo has been able to process more than 3000 transactions per second for hours on end. The blockchain is even capable of processing more than 5500 transaction per second during peak times.

Cost prediction
Due to the nature of Smilo’s Gas mechanism it will be a straightforward process to calculate the running costs of using the Smilo network for your corporation or business. Since every transaction on the Smilo network will be free of charge, there will not be any uncertainty about the running costs.

Smilo (XSM)

Smilo tokens are a necessary element — a license — for operating the distributed Smilo Platform. It is a form of licensing made by the clients of the platform to the machines executing the requested operations. To put it another way, Smilo tokens are the incentive ensuring that developers write quality applications since inefficient code will require more Gas. Additionally, it will ensure that the network remains healthy as users will be compensated for their contributed resources.

Smilo token creation
After the launch of mainnet, the speaker, also known as the block generating masternode, will be rewarded if they successfully generate a new block. For more information about the reward, please read the chapter ‘Masternode block reward’. Furthermore, the Smilo Platform will NOT pre-mine any Smilo tokens.

SmiloPay (XSP)

SmiloPay tokens are a currency which is issued to users who own Smilo tokens, SmiloPay tokens cannot be traded and can only be consumed, representing the fee to use network resources, such as: transactions, computing, storage and bandwidth. For every Smilo token you own, you will receive a certain amount of SmiloPay, and each SmiloPay token represents a share of the network resources.

In other words, the free transactions does not mean that the network can be used without any costs or limitations. The users of the network will still have to pay a fee, but this fee will be paid with time. The users will have to own Smilo tokens in order to generate free transactions, and the amount of owned Smilo tokens will determine the amount of transactions per second that can be executed by said user.

The usability of SmiloPay on the Smilo network can be compared to a battery. The amount of Smilo tokens a user owns determines the maximum capacity volume and the charging speed of the battery. Thus, if a user owns more Smilo tokens, the user will be able to store more SmiloPay tokens while accumulating them quicker. If the user then decides to use the device, it will consume a specific amount of energy, and this amount will be related to the capabilities of the device. So, depending on the necessary effort for the transaction, the network will use a specific amount of SmiloPay.

The SmiloPay mechanics differ from Ethereum's Gas mechanics as follows: for every account, the maximum amount of SmiloPay is fixed if the Smilo token balance of the account remains the same. SmiloPay tokens will be restored over time, and the recovery speed is positively related to the amount of Smilo tokens in the account. However, the operation of consuming SmiloPay on the Smilo blockchain is similar to the accumulation process that consumes gas in Ethereum. Broadcasting a transaction on the Smilo blockchain requires an account to consume SmiloPay tokens, the larger the amount of data carried during the transaction, the more SmiloPay tokens are consumed. Additionally, if the calculations of the data in the contract are more difficult to process it will consume more SmiloPay tokens. Furthermore, the order of transactions in the transaction pool is sorted by gas price from high to low.

SmiloPay properties

As mentioned before, the Smilo account has two properties regarding SmiloPay. Both these properties depend on the amount of Smilo tokens in the account:

- There will be a maximum amount of SmiloPay tokens.
- The recovery speed of the SmiloPay tokens.

For an overview of the SmiloPay properties, please visit Appendix A: Balance - SmiloPay comparison table.

Calculate SmiloPay balance

The calculation of SmiloPay on an account:

SmiloPay = Min(MaxSmiloPay, BlockGap * RecoverySpeed)

Where BlockGap is:

BlockGap = 'current block height' – 'the height of the last trading block on this account'

Calculate SmiloPay consumption by a transaction

The transaction Fee can be calculated by:

*SmiloPay = Gas * GasPrice*

For example, a regular transaction cost 21000 Gas and the Gas price is 5 Gwei. (5Gwei = 0.000000005 Smilo). Then a normal transaction will cost:

*TxFee = 21000 * 0.000000005 = 0.000105 SmiloPay*

<u>The maximum amount of SmiloPay</u>

Each account has a 'SmiloPay Maximum', based on the amount of Smilo the account holds.

$$MaxSmiloPay = (0.001 + (\sqrt{balance}/50000)) * 5$$

<u>The recovery speed of SmiloPay</u>
The recovery speed of the SmiloPay tokens will also depend on the amount of Smilo tokens in the account. The recovery speed can be calculated by:

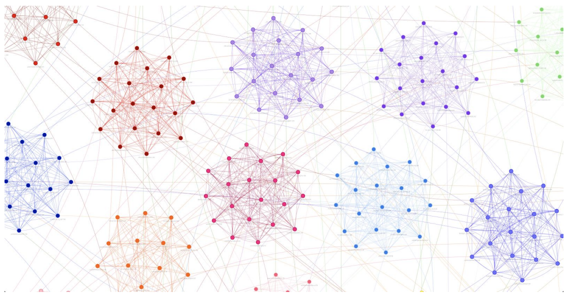
$$RecoverySpeed = (0.000001 + (\sqrt{balance}/750000)) * 0.5$$

Smilo Network Protocol (SNP)

The Earth orbits the Sun in our Solar System. The Sun is one star among the billions of others in the Milky Way Galaxy, and the Milky Way Galaxy is one galaxy among the billions of others in the Universe.

In the above example, each masternode is known as either a planet, such as the Earth, or as a Sun. Each planet is a part of a galaxy where everyone collaborates to create the next block of the blockchain. The clients, such as a wallet or explorer, are connected to one or multiple galaxies, creating a universe, and in this universe, the different galaxies share transactions and blocks to verify and update the blockchain.

This will keep the Smilo Network Protocol Fast, Secure, and Scalable. No matter how many galaxies exist.



<u>The Smilo Universe</u>
- Each sun is a masternode
- Each planet is a masternode
- The first masternode in a galaxy is the sun
- Each galaxy contains 1-19 planets and 1 sun
- Each planet communicates with 2 galaxies
- Each masternode in a galaxy will cooperate to create the next block
- Cheating galaxies will be punished
- There is no limit on the amount of galaxies
- Clients/ light-nodes are part of the universe and will verify the galaxies

<u>Virtual networks and network traffic decrease</u>
The masternode has 3 virtual networks:
- Galaxy A
- Galaxy B
- Universe (clients/ wallets/ etc.)

To decrease network traffic, blocks and transactions will by default only be broadcasted once. If an unknown block/message is received from virtual network Galaxy A, it will only be broadcasted to Galaxy B and clients on the Universe. If an unknown block/message is received from virtual network Galaxy B, it will only be broadcasted to Galaxy A and clients on the Universe. If a known block (a duplicate) is received, nothing will be broadcasted. Therefore the Smilo network decreases traffic by 50 percent.

Smilo Proof of Resources and Time (SPoRT)

Selecting a speaker

SPoRT is an acronym for Smilo Proof of Resource and Time. The SPoRT challenge determines which speaker (fullnode) will generate the upcoming block. After the generation of a block, the SPoRT challenge will start over to determine the next speaker that may generate the upcoming block.

- A network cannot speak two times in a row
- When a node is a speaker, it cannot speak for the next four blocks

Masternode blacklisting

All masternodes are untrusted by default, but the purpose of a masternode is to secure the network. Therefore each node can blacklist bad Actors. There are three requirements for a masternode to be blacklisted from the network:

- The hash is valid
- The signature is valid
- One or more transactions are invalid (which will result in invalid blocks)

If the hash or signature are invalid, the block will never be broadcasted. When a speaker commits a new block, the block is validated by all the masternodes in the galaxy. If the hash or signature is invalid, there is a chance that a masternode has tampered with the block. If the hash and signature are valid, we know the block is generated by the speaker, and therefore the transaction can be validated. If there are any invalid transactions while the source is validated, we will add the speaker to the blacklist and the masternode will be punished.

Furthermore, all the nodes that approved the invalid block will be punished and blacklisted by the node. This will result in a split, where the blacklisted masternodes will become orphans.

Masternode block reward

Masternodes which support the network by generating blocks (by becoming a speaker) will be rewarded as follows:

From Block	To Block	Block Reward
1	20.000.000	4
20.000.001	40.000.000	2
40.000.001	60.000.000	1,75
60.000.001	80.000.000	1,5
80.000.001	100.000.000	1,25
100.000.001	120.000.000	1,0
120.000.001	140.000.000	0,8
140.000.001	160.000.000	0,6
160.000.001	180.000.000	0,4
180.000.001	200.000.000	0,2
200.000.001	400.000.000	0,1
400.000.001	800.000.000	0,05
800.000.001	1.600.000.000	0,025
1.600.000.00	3.200.000.000	0,0125

In total, 350 million Smilo tokens will be gradually accumulated over the course of the next 100+ years. Furthermore, during the first 200 million blocks, the relative inflation will decrease every 20 million blocks. After the first 200 million blocks, the block reward will half each doubling of blocks.

Smilo BFT+

The Smilo BFT+ algorithm will be implemented in various different stages. By implementing the algorithm in these stages, Smilo is more confident to deliver our software without any hesitation.

In the first stage, Smilo's platform will be an effective and secure proof-of-authority (PoA) protocol based on Ethereum's Clique implementation. However, Smilo's implementation will add important features, such as: Round, Fullnodes, Speaker, and Fullnode elections/ evictions.

Initially, during the first stage of the development of the Smilo BFT+ algorithm, there will only be two kinds of nodes: Full nodes (masternodes) and normal nodes. Full nodes are responsible for creating the blocks of the blockchain. The blocktime will be between one to ten seconds, with a single confirmation (no regular forks). At the beginning, Full nodes will be configured on a permissioned network. However, this permissioned network is temporary and will be lifted as soon as the protocol becomes more mature. The BFT+ consensus can tolerate at most *F* - Byzantine or Faulty nodes in *N* - Fullnodes. F = (N) / 3.

Within the algorithm, a new round start every time a block is committed by the (2F+E) Full nodes in the blockchain. Before each block creation, an election takes place to choose the Speaker. The Speaker is then responsible to create the block. The block will be validated, and it will only be approved if 66% of the Full nodes approve it.

During this first stage of development, only the Round Robin algorithm will be supported for the selection of the speaker. A Full node can vote to elect a new candidate or evict an elected Full node. The network requires 2F+E Full nodes agreeing to it. The voting process makes use of APIs exposed to Web3.js

During the second stage of the implementation, Smilo will implement the Smilo network Protocol (SNP), Masternode blacklisting, and the Smilo Proof of Resources and Time (SPoRT).

Smilo BFT+ benefits

Single chain removes forking

There is always only one block being proposed. The single chain removes forking, meaning that there is no need for uncle blocks and the transaction can't be undone on the chain at a later time.

Lower effort to construct and validate blocks

The effort to create a block is significantly reduced. The Smilo BFT+ algorithm reaches 5000 transactions per seconds at peak times and it is able to process 3000 transactions per second for extended periods of time.

Super majority of nodes to validate a block

66% of the masternodes are required to sign the block prior to the insertion to the chain. The block signers are always rotating, thus ensuring that a malicious node can't influence the chain for a long time.

Blocktime

Blocks will be created between one to ten seconds and with a single confirmation (no regular forks). To protect the network, empty blocks are not allowed to be mined.

Performance

Where other blockchain platforms struggle with a large amount of transactions, Smilo platform is able to handle more than 3300 transactions per second on average and more than 5,000 transactions per second at peak. Our consensus protocol consists of the Smilo BFT+ and the Smilo Network Protocol, which allows us to easily implement sharding on our blockchain.

This amount of transactions per second makes the

Smilo platform ready for the future, where we expect large amounts of transactions to occur.

Watch our performance test results here: https://www.youtube.com/watch?v=-vNawAjdNAw

Smilo Private Vault (SPV)

(Private Smart Contracts)

The Smilo Private Vault (SPV) is P2P software that stores and shares your private smart contract state across the network. It is a software program written in Golang that makes use of the Ethereum P2P capabilities to find peers and transfer encrypted data.

The Smilo Private Vault is created to encrypt and share private transactions via P2P, outside of the blockchain. The protocol will search the network and identify the Vault peer that matches with the peer that is supposed to receive the data at the time of creating the transaction. During this transaction, the smart contract state will never be saved on the blockchain, only a hash that is consequently used as a checksum will be stored on the blockchain. With this hash, nodes can validate whether they have a received a valid state or not. A not valid state is automatically refused and the peer who sent the faulty data is blacklisted temporarily.

Public smart contracts cannot interact with Private smart contracts, as private smart contracts are only known to certain entities. However a private smart contract can make use of a public smart contracts information to change its own state. But will never be able to change the Public smart contract's state. The data that is sent via P2P to a node, is encrypted using TweetNACL "crypto library in a 100 tweets" in Golang. Ref: http://nacl.cr.yp.to/secretbox.html

The Smilo Private Vault calculates a 32 byte shared key for the hashed key-exchange described for curve 25519 and uses this key plus a random nonce to encrypt and secure the smart contract state. Ref. http://nacl.cr.yp.to/box.html

TweetNACL was audited and the source code was found to be bug-free. Ref: https://tweetnacl.js.org/audits/cure53.pdf

Smilo Private Vault zk-SNARKS (SPV-Z)

(Private Smart Contracts zk-SNARKS)

Zk-SNARKs were introduced by Zcash in the Zcash protocol as a zero-knowledge cryptography. The acronym zk-SNARK is an abbreviation for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" and refers to a proof construction where one can prove possession of certain information, e.g. a secret key, without revealing that information, and without any interaction between the prover and verifier. "Zero-knowledge" proofs allow one party (the prover) to prove to another party (the verifier) that a statement is true, without revealing any information beyond the validity of the statement itself. For example, given the hash of a random number, the prover could convince the verifier that there indeed exists a number with this hash value, without revealing what it is. "Succinct" zero-knowledge proofs can be verified within a few milliseconds, with a proof length of only a few hundred bytes, even for statements about programs that are very large.

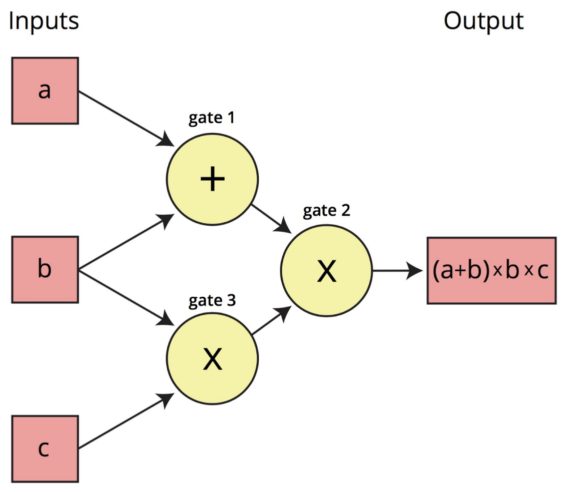
In the first zero-knowledge protocols, the prover and verifier had to communicate back and forth for multiple rounds, but in "non-interactive" constructions, the proof consists of a single message sent from prover to verifier. Currently, the only known way to produce zero-knowledge proofs that are non-interactive and short enough to publish to a blockchain, is to have an initial setup phase that generates a common reference string shared between prover and verifier. We refer to this common reference string as the public parameters of the system.

In order to have zero-knowledge privacy, the function determining the validity of a transaction according to the network's consensus rules must return the answer of whether the transaction is valid or not, without revealing any of the information it performed the calculations on. This is done by encoding some of the network's consensus rules in zk-SNARKs. At a high level, zk-SNARKs work by first turning what you want to prove into an equivalent form about knowing a solution to some algebraic equations. In the following section, we give a brief overview of how the rules for determining a valid transaction get transformed into equations that can then be evaluated on a candidate solution without revealing any sensitive information to the parties verifying the equations.

Computation → Arithmetic Circuit → R1CS → QAP → zk-SNARK

The first step in turning our transaction validity function into a mathematical representation is to break down the logical steps into the smallest possible operations, creating an "arithmetic circuit". Similar to a boolean circuit where a program is compiled down to discrete single steps, such as: AND, OR, and NOT, when a program is converted to an arithmetic circuit, it's broken down into single steps consisting of the basic arithmetic operations of addition, subtraction, multiplication, and division (although in our particular case, we will avoid using division).

Here is an example of what an arithmetic circuit looks like for computing the expression (a+b)*(b*c):



An important aspect of our solution is its autonomy. The sender is not required to cooperate with other users or a trusted third party to complete transactions, hence each participant produces a transaction independently.

SPV-Z is implemented in Golang using libsnark, a C++ library for zk-SNARK proofs. It implements zk-SNARK schemes, which are a cryptographic method for proving/ verifying, in zero knowledge, the integrity of computations.

Smilo Masternodes

The masternode, a concept derived from Dash's full-node servers, exist at the necessary service facility to ensure that the blockchain provides certain services and basic performance. In the Dash network, the masternode operates based on the Proof of Stake (PoS) mechanism and forms a conceptual double-layer network together with the miner node responsible for completing the Proof Of Work mechanism (PoW).

Under the Smilo BFT+ consensus mechanism, the masternode group replaces the miner role in the PoW mechanism to jointly handle transaction verification, and broadcast work. The Smilo masternode server requirements meet general cloud services, and the node server requirements allow the network to be more decentralised. To become a Smilo masternode you will need to hold 20.000 Smilo and set up a fast enough server to handle all the requests. If a Masternode cannot keep up with the other masternodes, it will not generate blocks, but it will be active as a network validator.

Masternode responsibilities
The responsibilities of the masternode include:

- Transaction Verification:
  Verify the signature of the transaction, account balance etcetera, execute the transaction and smart contract, and using legitimate transactions to perform block generation.
- Community autonomy:
  The community has voting rights on the proposals, meaning that the proposal will reflect the community's discussions. This will involve all aspects of Smilo’s development, including but not limited to the direction of technical iterations, operation plan adjustments, changes in economic parameters, and so on.

Initially, to become a Masternode on the Smilo chain, the individual or entity voluntarily discloses who they are (identity and reputation by extension) by submitting to a (KYC) procedure and being able to satisfy the minimum requirements set by the Foundation in exchange for the right to validate and produce blocks.

Operating masternodes
Requirements** for operating a masternode:

- Hold 20.000 Smilo
- The masternode should have an independent IP address
- At least 16 GB of RAM
- Minimum 1TB of SSD space
- Intel® Xeon® Processor 2xE5-2670 or equal/better

Attacks

The Smilo blockchain platform has been designed with a high level of security in mind. The Smilo BFT+ algorithm combines speed, scalability and a high level of protection.

66% Attack
The existence of the masternode requirements make it extremely expensive to initiate an attack based on building a large number of nodes. As an example, if the total number of masternodes equals 3000, the attacker would need to control or create 4000 additional masternodes in order to obtain an attack success rate of 1%. This means that the attacker has to purchase 80 million Smilo tokens, which is close to one-third of the total token circulation.

So, it takes a huge effort to attack the network with only a small probability of success. Combine this with the locked Smilo tokens and the punishment for bad masternodes, the overall possibility of attack is reduced, and this kind of attack becomes even more unrealistic.

Double spending Attack
Bitcoin prevents double-spending problems through PoW and block confirmation. Due to its design limitations (the transaction confirmations), the network needs to ‘wait’ for a longer time.
Smilo adopts the same scheme as ETH, to reject double-spending transaction, using the Nonce value. This kind of scheme description is one account’s transaction in the network and are executed based on the sequence of transaction.
This scheme is also used to cancel pending transactions which are not being processed for a long time and also provides the possibility to carry out a transaction by setting a higher Gas Price to replace the pending transaction which has the same Nonce value.

Sybil Attack
This refers to an attack that benefits from creating multiple accounts on the network.

The single-account transaction attenuation ability, constructed by the SmiloPay mechanism, can only result in an effective sybil attack by holding a large amount of Smilo. Nevertheless, the trading pool which runs the transaction order, based on the Gas Price, will further reduce the impact of the attack.

DDoS Attack

A DDoS attack refers to a large number of spam requests to the host server in a short period of time, which may cause some of the master nodes to go offline and can result in service disruptions. Because of the Gas and SmiloPay mechanisms, DDoS-type attacks only have a small influence on the Smilo network.

In Smilo, the number of transactions that can be created by an account are limited by the account's balance in XSM. This mechanism results in a high cost of launching a large quantity of transactions. The detailed design below explains more:

1) Account threshold: An account with fixed balance can only send an X amount of transactions per block due to SmiloPay rules.
2) Capacity: The higher amount of Smilo an account has, the more transactions per block can be send.
3) Ordering: The higher amount of Smilo an account has, the more it can pay for SmiloPay, consequently more transactions will be selected to be in a block.
4) Contract min balance: A contract can only be called by other accounts when its balance is greater than 100 XSM.
5) Maximum stack size: In order to prevent a infinite loop to occur, the max. Stack size is limited to 1024 (could be changed in near future)

Finney Attack

The Finney attack is named after a Bitcoin user, Hal Finney. It's an attack that exploits unconfirmed transactions in Bitcoin to fraudulently accept bitcoin payments, a variation of a double-spending attack. The precondition for this attack is that the merchant trusts the unconfirmed transaction and immediately ships the shipment after receiving the unconfirmed transaction and cannot be revoked. This is actually using the time difference of the high-latency transaction confirmation service such as found in BTC. In Smilo however, the near real-time transaction greatly reduces the possibility of this attack.

Long range attack

A long-range attack is when the attacker goes back to an old block, generates a new Blockchain branch and broadcasts the branch to try to override the existing trunk. The fabricated branch is usually much longer than the trunk and manages to fool the consensus protocol.

Long-range attacks cannot be used to attack our protocol. Due to the Δ-second interval
between consecutive blocks, it is impossible to produce a longer chain.

Our consensus chooses the trunk based on voting, therefore the attacker needs to own 66% of the masternodes. The attack would then become a 66% attack which has been described above.

Conclusion

Smilo combines the best of blockchains with a safe and reliable "free from transaction costs", making the large and complex smart contracts economically viable, stable and continuous. The Smilo network protocol is highly scalable and gives real-time transaction feedback realised by the master node network.

The large number of users will have an excellent interactive experience, which will change the impression that the blockchain transaction confirmation waiting time is too long. The blockchain industry is still in its early stages and is struggling to face the technological challenges. Only by building on the well-established technical approaches, step by step, and with a deep understanding about the industry, Smilo can reduce the overall risks-faced and complete the long-term goal of becoming the mainstream blockchain application platform.

The limitations of current technology will limit the popularity of blockchains in daily life, and price-speculations will continue for a long period of time. Smilo will not forget their original objective; improving the blockchain technology, exploring the application in various industries as its own responsibility, using decentralized technology and ideas to improve social operation efficiency, reducing the cost of social operations, and contributing to achieve a fairer society.

Ethereum

Ethereum is currently one of the most advanced blockchains. Some of the great ideas have included the introduction of an account model such that the “state” in the fundamental transaction-based state machine model can store information not restricted to the balance information; the concept of a “smart contract” that allows blockchain to describe more complicated objects and activities in the real world through consensus-based computations and the invention of the Ethereum Virtual Machine (EVM) and the EVM code that enables smart contracts.

Despite being a major technological milestone, Ethereum has not been made suitable for hosting large-scale commercial decentralized applications (DApps) that could engage our day-to-day activities. One of the main reasons is that there hasn’t been an effective governing structure set up, from the very beginning, for Ethereum to allow efficient and private transitions. Secondly, Ethereum lacks a suitable economic model to allow enterprises to run their DApps with a controllable and predictable cost. Considering the level of volatility of the Ether price, it is almost impossible for companies to predict the future price of Ether or the cost of running a DApp based on Ethereum for a period of time.

The Smilo Blockchain is designed to tackle the above problems. It does not just provide pure technical solutions but is empowered by novel governance and economic models, which, we believe, will push forward broader blockchain adoption and the creation of ecosystems with more trust and efficiency. The Smilo Blockchain is built from scratch, but adopted some of the essential building blocks of Ethereum (e.g., the account model, EVM, modified Patricia tree, Web3.js and RLP encoding method). But most importantly, it is packed with technical features that are tailor made for the actual needs of both enterprise and individual users. We believe that the new features give both users and developers added flexibility and powerful tools to achieve their goals
on the Smilo Blockchain.

Acknowledgements

Special thanks to Nickel van de Mortel, Michael Hassan, Jesper Winkelhorst, Fabio Cruz and Cristiano Mazzon for reviewing this technical paper. For the good input, corrections on bad language and the many reads.

Satoshi Nakamoto, Thank you for creating this amazing world changing technology we call Blockchain, and for giving it freely to the world. Thank you for giving us the tools we need to free ourselves (if only a little) from untrusted third parties.

Vitalik Buterin, Thank you for your pure approach to blockchain. Thank you for truly caring and calling out the greed and the scams. Keep up the good work!

References

- Ethereum White Paper. (2017, 18 September). Retrieved on 30 November 2017, from https://github.com/ethereum/wiki/wiki/White-Paper
- Smart Contracts: The Blockchain Technology That Will Replace Lawyers. (2017, 11 December). Retrieved on 15 December 2017, from http://www.machinelearningto.com/blog/smart-contracts-the-blockchain-technolog Y-that-will-replace-lawyers
- Goldwasser, S., Micali, S., & Rackoff, C. (1989, February). The knowledge complexity of interactive proof systems. Retrieved on 25 March 2018, from http://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Proof%20System s/The_Knowledge_Complexity_Of_Interactive_Proof_Systems.pdf
- Smilo whitepaper. (2018, January). Retrieved on 21 February 2018, from https://smilo.io/files/Smilo_White_Paper_Latest.pdf

ABOUT US

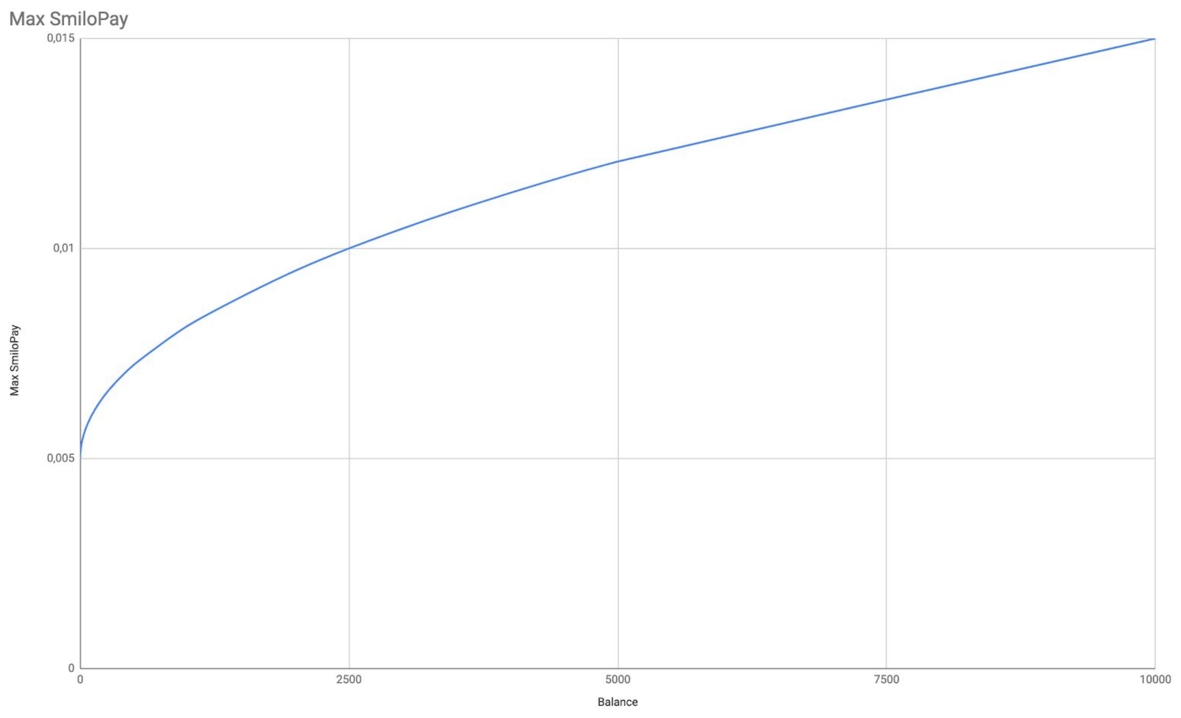
Smilo Platform is a unique full-featured hybrid blockchain platform that will be able to facilitate hybrid transactions, hybrid smart contracts and hybrid decentralised applications — with ‘hybrid’ referring to both public and private. Smilo uses their unique blockchain technology to facilitate an alternative protocol for decentralised applications, including GDPR compliant private vaults, infinite scaling and accurate cost predictions.

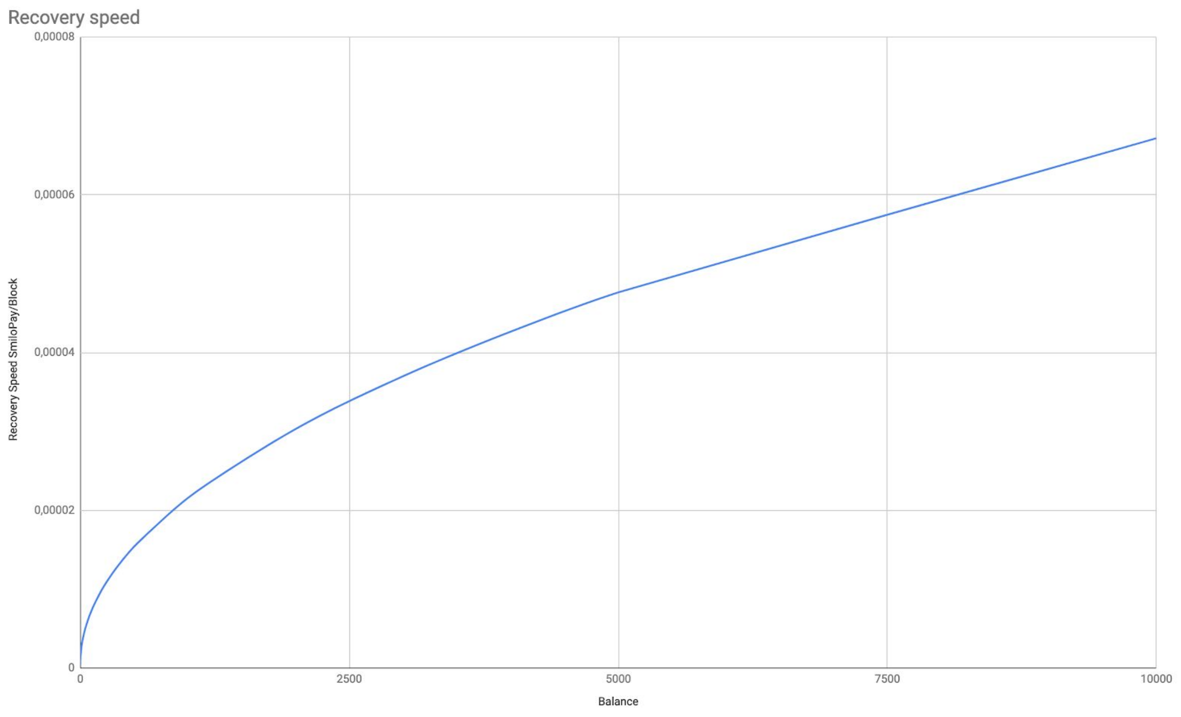
Appendix A: Balance - SmiloPay Comparison table

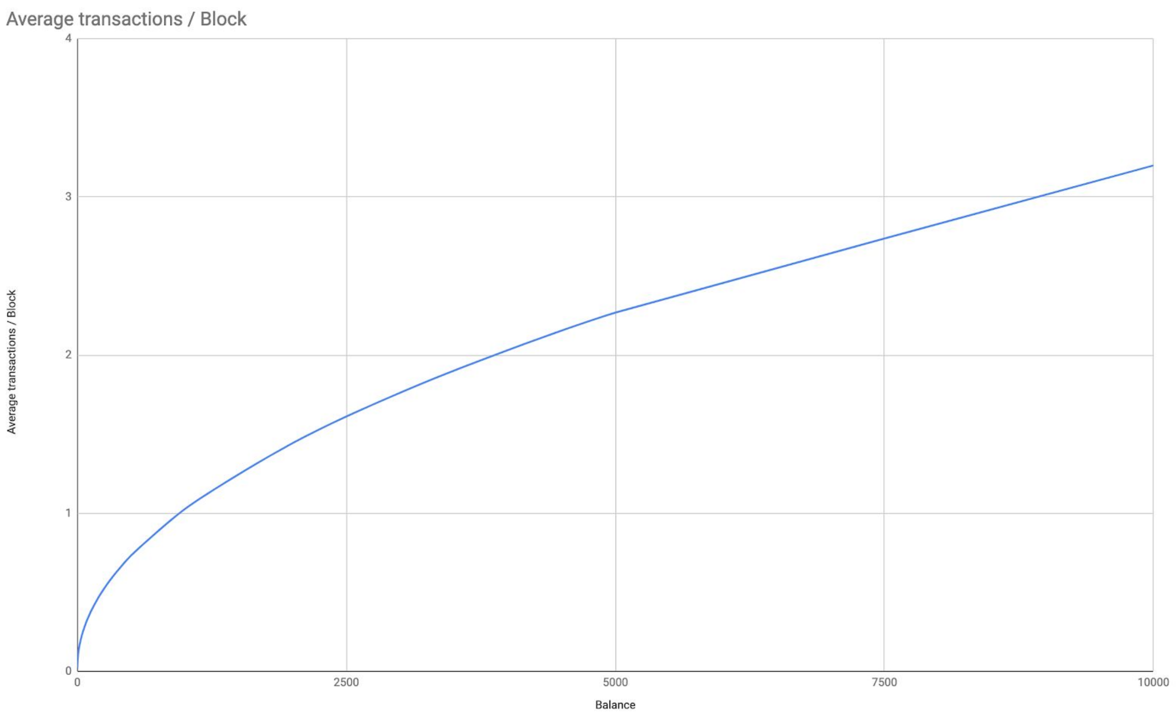
Below table can be used as a basic reference. The maximum and average transactions per block calculations are based on a GasPrice of 0.000000001 (1 Gwei)

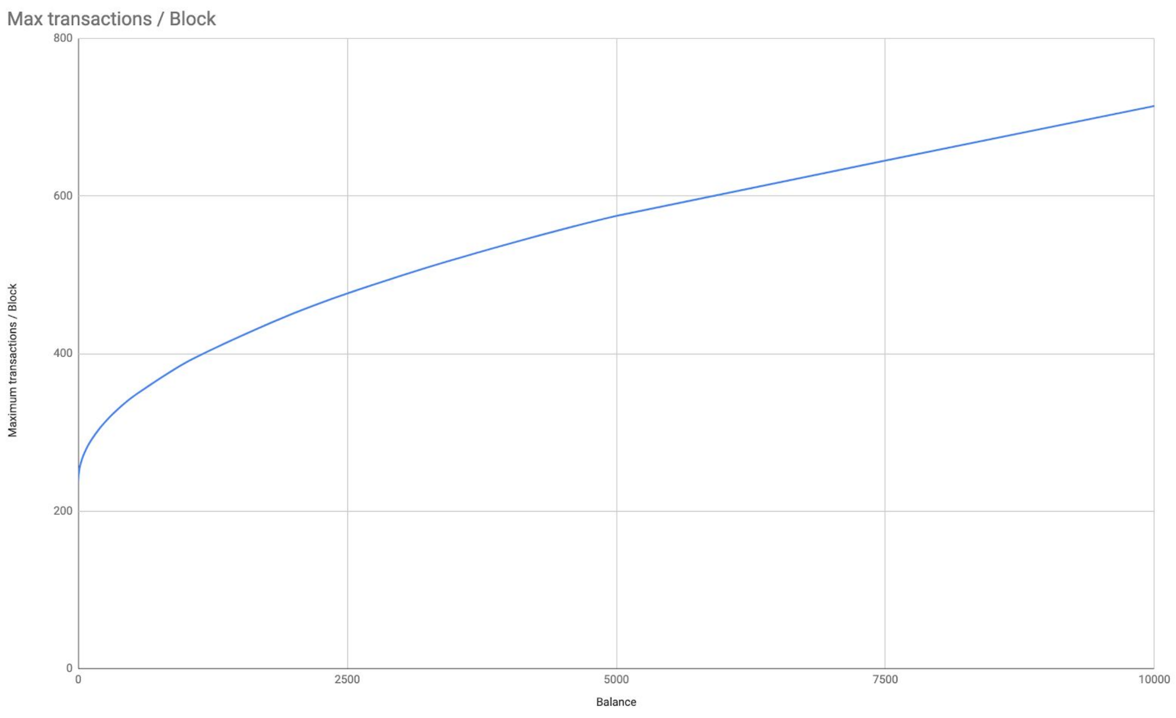
Balance	**MaxGas**	**MaxSmiloPay**	**Recovery Speed**	**Max TX per Block**	**Avg TX per Block**	**Full Charge duration**
(Smilo)	(GAS)	(SmiloPay)	SmiloPay/Block			Blocks
0,1	5031620	0,00503162	0,00000071	239,60095	0,03381	7086,788732
1	5100000	0,0051	0,00000117	242,85714	0,05571	4358,974359
10	5316230	0,00531623	0,00000261	253,15381	0,12429	2036,869732
100	6000000	0,006	0,00000717	285,71429	0,34143	836,8200837
500	7236070	0,00723607	0,00001541	344,57476	0,73381	469,5697599
1000	8162280	0,00816228	0,00002158	388,68	1,02762	378,2335496
5000	12071070	0,01207107	0,00004764	574,81286	2,26857	253,3809824
10000	15000000	0,015	0,00006717	714,28571	3,19857	223,3139795
100000	36622780	0,03662278	0,00021132	1743,9419	10,06286	173,3048457
1000000	105000000	0,105	0,00066717	5000	31,77	157,3811772

For reference only, production values might change according to Gas Price.


Max SmiloPay
0,015
0,01
0,005
0
Max SmiloPay
0
2500
5000
7500
10000
Balance


Recovery speed
0,00008
0,00006
0,00004
0,00002
0
Recovery Speed SmiloPay/Block
0
2500
5000
7500
10000
Balance


Average transactions / Block
Average transactions / Block
Balance
0
1
2
3
4
0
2500
5000
7500
10000


Max transactions / Block
Maximum transactions / Block
Balance
0
200
400
600
800
0
2500
5000
7500
10000

Appendix B: Fee schedule

Value	Mnemonic	Gas Used	Removed from stack	Added to stack	Notes
0x00	STOP	0	0	0	Halts execution.
0x01	ADD	3	2	1	Addition operation
0x02	MUL	5	2	1	Multiplication operation.
0x03	SUB	3	2	1	Subtraction operation.
0x04	DIV	5	2	1	Integer division operation.
0x05	SDIV	5	2	1	Signed integer division operation (truncated).
0x06	MOD	5	2	1	Modulo remainder operation
0x07	SMOD	5	2	1	Signed modulo remainder operation.
0x08	ADDMOD	8	3	1	Modulo addition operation.
0x09	MULMOD	8	3	1	Modulo multiplication operation.
0x0a	EXP	FORMULA	2	1	Exponential operation.
0x0b	SIGNEXTEND	5	2	1	Extend length of two’s complement signed integer.
0x10	LT	3	2	1	Less-than comparison.
0x11	GT	3	2	1	Greater-than comparison.
0x12	SLT	3	2	1	Signed less-than comparison.
0x13	SGT	3	2	1	Signed greater-than comparison.
0x14	EQ	3	2	1	Equality comparison.
0x15	ISZERO	3	1	1	Simple not operator.
0x16	AND	3	2	1	Bitwise AND operation.
0x17	OR	3	2	1	Bitwise OR operation
0x18	XOR	3	2	1	Bitwise XOR operation.
0x19	NOT	3	1	1	Bitwise NOT operation.
0x1a	BYTE	3	2	1	Retrieve single byte from word
0x20	SHA3	FORMULA	2	1	Compute Keccak-256 hash.
0x30	ADDRESS	2	0	1	Get address of currently executing account.
0x31	BALANCE	400	1	1	Get balance of the given account.
0x32	ORIGIN	2	0	1	Get execution origination address.
0x33	CALLER	2	0	1	Get caller address.

0x34	CALLVALUE	2	0	1	Get deposited value by the instruction/transaction responsible for this execution.
0x35	CALLDATALOAD	3	1	1	Get input data of current environment.
0x36	CALLDATASIZE	2	0	1	Get size of input data in current environment.
0x37	CALLDATACOPY	FORMULA	3	0	Copy input data in current environment to memory.
0x38	CODESIZE	2	0	1	Get size of code running in current environment.
0x39	CODECOPY	FORMULA	3	0	Copy code running in current environment to memory.
0x3a	GASPRICE	2	0	1	Get price of gas in current environment.
0x3b	EXTCODESIZE	700	1	1	Get size of an account’s code.
0x3c	EXTCODECOPY	FORMULA	4	0	Copy an account’s code to memory.
0x40	BLOCKHASH	20	1	1	Get the hash of one of the 256 most recent complete blocks.
0x41	COINBASE	2	0	1	Get the block’s beneficiary address.
0x42	TIMESTAMP	2	0	1	Get the block’s timestamp.
0x43	NUMBER	2	0	1	Get the block’s number.
0x44	DIFFICULTY	2	0	1	Get the block’s difficulty.
0x45	GASLIMIT	2	0	1	Get the block’s gas limit.
0x50	POP	2	1	0	Remove item from stack.
0x51	MLOAD	3	1	1	Load word from memory.
0x52	MSTORE	3	2	0	Save word to memory
0x53	MSTORE8	3	2	0	Save byte to memory.
0x54	SLOAD	200	1	1	Load word from storage
0x55	SSTORE	FORMULA	1	1	Save word to storage.
0x56	JUMP	8	1	0	Alter the program counter
0x57	JUMPI	10	2	0	Conditionally alter the program counter.
0x58	PC	2	0	1	Get the value of the program counter prior to the increment corresponding to this instruction.
0x59	MSIZE	2	0	1	Get the size of active memory in bytes.
0x5a	GAS	2	0	1	Get the amount of available gas, including the corresponding reduction for the cost of this instruction.
0x5b	JUMPDEST	1	0	0	Mark a valid destination for jumps
0x60 -- 0x7f	PUSH*	3	0	1	Place * byte item on stack. 0 < * <= 32
0x80 -- 0x8f	DUP*	3	*	* + 1	Duplicate *th stack item. 0 < * <= 16

0x90 -- 0x9f	SWAP*	3	* + 1	* + 1	Exchange 1st and (* + 1)th stack items.
0xa0	LOG0	FORMULA	2	0	Append log record with no topics.
0xa1	LOG1	FORMULA	3	0	Append log record with one topic.
0xa2	LOG2	FORMULA	4	0	Append log record with two topics.
0xa3	LOG3	FORMULA	5	0	Append log record with three topics.
0xa4	LOG4	FORMULA	6	0	Append log record with four topics.
0xf0	CREATE	32000	3	1	Create a new account with associated code.
0xf1	CALL	FORMULA	7	1	Message-call into an account.
0xf2	CALLCODE	FORMULA	7	1	Message-call into this account with an alternative account’s code.
0xf3	RETURN	0	2	0	Halt execution returning output data.
0xf4	DELEGATECALL	FORMULA	6	1	Message-call into this account with an alternative account’s code, but persisting the current values for sender and value.
0xfe	INVALID	NA	NA	NA	Designated invalid instruction.
0xff	SELFDESTRUCT	FORMULA	1	0	Halt execution and register account for later deletion