Main components

So we're developing a hybrid consensus that works on the following notions:

1. A permissioned BFT that runs on a few nodes in the permissionless POW based network.
2. The BFT committee is a rotating one, preventing corruption in a timely manner
3. The BFT committee is responsible for transaction validation, and the POW nodes are only responsible for choosing/electing the committee members according to some rules we've derived and re-defined, as mentioned in the yellow paper!
4. The new permissioned VM, we've surmised, could be inspired from the EVM, but with different block states and transaction execution flows
5. The contemporary permissionless EVM in the POW chain co-exists with this new permissioned VM (which we call Truechain Virtual Machine - TVM)
6. The TVM would be the one, as per current discussions, to validate any transaction, while EVM would need to be re-worked to not really mine for consensus, but for election of BFT and maybe lightweight wallet transactions (the part we'd be happy to discuss with you in design)
7. The incentivation model needs to be re-worked such that it is based off of TVM, and also, that we somehow still reward the miners in POW chain.
8. We would eventually support sharding for the BFT committee nodes, for scalability.
9. A compensation infrastructure, which accounts node spec non-uniformity (as in different CPU/memory/network bandwidth in the node pool), would eventually be a part of the consensus, thus speeding up transactions.
10. The smart contracts execution would thus only happen in TVM (BFT node) and we do think the deployment should be supported from POW nodes, thus raising a question about state replication in a hybrid setting.

Notes:

- We might need changes in solidity as well as the current Mist wallet in Ethereum, which is what we chose as our framework.

Timeline

#TODO: add details for specs/pseudo code specifics in timeline

Quarter	**Date (onwards or exact)**	**Phases**	**Details**	**Remarks**
Q1/18		**Project.__init__()**	V <3 Python	V also <3 GoLang
Q1/18		Whitepaper release		https://www.truechain.pro/EnTruechain.pdf
Q2/18		Yellow Paper - First Draft	• Hybrid consensus on arXiv	https://arxiv.org/abs/1805.01457
			Basic TVM	EVM (POW) + TVM (BFT) ==> PBFT + POW on single node
			Incentivization scheme	Develop an incentivisation scheme considering a different Txn fee model for the new hybrid virtual machines combination.

		Yellow Paper TIP - 1		
			Update arXiv paper	
			hybrid consensus pseudo code release	
			decoupling and coupling / rearrange ethereum to fit the consensus	
			adjust ethereum GAS token economy model	
		Coding	Implement hybrid consensus codebase	
		(to be reordered, shortly and broken down further according to chronological importance and dependencies)		Node init: 1. `forkVirtualNodeBFT()` 2. `SpawnBFTnode()`
				Validation: 1. `onChainValidation()` a. `validateFromPOW()` b. `FetchTxnHistory()` 2. `reElect()` 3. `checkTxnOrder()` 4. `updateSnailchain()`
				For view change and node corruption detection/complaing: 1. `ViewChange()` 2. `complainToSnailChain()`
				For signatures: 1. `keygen()` 2. `SignLog()` 3. `Keccak256/ECDSA wrappers`
				Logging: 1. `dailyLogOutput()` 2. `CreateLOG()` 3. `createTxTuple()`
				Transaction pool: 1. `mempoolSubprotocol()` a. `propose()` b. `query()` c. `others..` 2. `DailyOffchainProtocol()` 3.
				Election and VRF: 1. `SeedSelectorVRF()` 2. `proposeStakeTransactions()`
				Snapshots:

				1. `createSnapShot()`
				Communication: 1. `gossipTxn()`
				Sharding and Compensation infrastructure:
				Raincheck for basic tests - Integration, functional and unit
			Geth re-engineering	
			Infra monitoring / devops	
			Security review	
			Server security hardening	
			Mist / Wallet re-engineering	
			frontend	
			Release design documents for smart contract execution	publicly on github/elsewhere
			Custom Dapps	
		Experimental release - r1 - v0.?		
			Developer Documentation for consensus	
		SCALE LAB: Setup public/private/hybrid cloud infra		
			Establish internal scale lab across global regions	
		Simulation and Testing		
			Testbed preparation	
			simulation basic for server load/cost	
			Adjustments, bug smashes and performance improvements	
		Experimental release - r2 - v0.?		
			TESTNET	
			Adjustments, bug smashes and performance improvements	

		Experimental release - r3 - v0.?		
Q3/18			Dapps dev on ethereum	
			MAIN NET	
		Requests for Feature Enhancements (RFEs) and TIPs (Truechain Improvement Proposals)		
			X86 considerations	
			And other future plans..	

January, February and March (Q1);
April, May and June (Q2);
July, August and September (Q3);
October, November and December (Q4).

Core consensus

On PBFT chain funcs

Phase 1:
#TODO: break this down further

Method	**Functionality**
dailyLogOutput()	outputs the log to a done(_,_) to be then sent over to non-member nodes
CreateLOG()	creates the tuple for daily log, based on logs of all other nodes, the tracking of signed logs is taken care of by non-members of committee
mempoolSubprotocol()	keeps a track of incoming transactions with a Union set. supports query method to return confirmed transactions
DailyOffchainProtocol()	conducts conditional election of a committee member, based on whether node is a BFT member or not.
keygen()	
forkVirtualNodeBFT()	
proposeStakeTransactions()	propose special stake in and stake out transactions to temporarily freeze their token assets based on Theta x csize (Theta is a manual parameter [0,1])
SpawnBFTnode()	takes following inputs from POW chain: - a historical log -

onChainValidation()	
	a historical log
complainToSnailChain()	Triggers re_elect()
ViewChange()	triggers a viewchange and in turn, complainToSnailChain()
createTxTuple()	(R, l, tx) R - current day, l is the nonce (sequence of txn in that day)
SeedSelectorVRF()	
gossipTx()	honest committee member then gossips the signed tuple to the networ
createSnapShot()	creates a world state snapshot

Phase 2:

- Timestamp verification (figure 1 in truechain yellow paper)
- create_shard() and speculative_transaction() (figure 2 in truechain yellow paper)
-

On POW chain funcs

Phase 1: fPOW (fruitchain)

- re_elect() - POW mining function, that elects new committee members using [Theta, LRU, stake_in/out] (csize number of blocks inside snailchain) based off of one of the following:
    - A view change due to corruption
    - A Kth day trigger from physical timestamp restriction
    - An expired Tstamp interval for which the node should be a DailyBFT committee member
- update_snailchain() - receives the done(_,_) hash from N PBFT nodes and is executed by each node on the chain to be then written on the snailchain ledger

Wallet engineering

-

Simulation

Determine the following:

- upper/lower bounds on shard sizes
- Lambda security parameter bounds
- Theta manual parameter

Test Suit

1. Checks for Consistency & Liveness

2. Checks for security
3. Test suit for corruption

4.

Smart Contract re-engineering / solidity

TVM / WASM

Documentation

Appendix

Notation

Variable	Meaning
`tx`	a transaction
`l`	sequence number of a transaction within each BFT instance
`LOG`	the totally ordered log each node outputs, LOG is always populated in order
`log`	log of one BFT instance, referred to as daily log
`log[l : l′]`	transactions numbered l to l′ in log
`log[: l]`	log[1 : l]
λ	security parameter
α	adversary's fraction of hashpower
δ	network's maximum actual delay
Δ	a-priori upper bound of the network's delay (typically loose)
`csize`	committee size, our protocol sets csize := λ
`th`	th := ⌈csize/3⌉, a threshold
`lower(R), upper(R)`	lower(R) := (R − 1)csize + 1, upper(R) := R · csize
`chain`	a node's local chain in the underlying snailchain protocol
`chain[: −λ]`	all but the last λ blocks of a node's local chain
`MinersOf(chain[s : t])`	the public keys that mined each block in chain[s : t]. It is possible that several public keys belong to the same node.
`{msg}pk−1`	a signed message msg, whose verification key is pk
`Tbft`	liveness parameter of the underlying BFT scheme