Ethereum could potentially open a few bird cages with one key: ETH-collateralized GAS tokens.

How TLDR

- There are 3 contracts:
- ERC20 GAS token contract
- Credit facility (CF) contract (~oracle-less govern-less auto-pilot Sai).
- Automatic market making (AMM) contract (~Uniswap V2).
- ERC20 GAS token contract
- Credit facility (CF) contract (~oracle-less govern-less auto-pilot Sai).
- Automatic market making (AMM) contract (~Uniswap V2).
- The base protocol accepts GAS tokens for fee payment

. This does not

undermine ETH enshrined status because GAS is ETH-backed (details below). If this EIP goes through, then buying gas from the AMM can become a typical 1st instruction (fill up on gas at the beginning of tx).

- Anyone can deposit

ETH into the CF to mint new GAS tokens up to a minimum over-collateralization threshold \alpha

% such that \alpha \times (\frac{GAS_minted}{ETH_deposited})\geq

a moving average (MA) GAS/ETH price. Setting \alpha

=125% is sufficient of a cushion because MA is smooth compared to the spot price. In contrast \alpha

is 133% and 150% on Compound and MakerDao, respectively.

- Minters can pool

GAS:ETH in the AMM and collect fees from traders/users (who “load up on GAS” at the beginning of the tx JIT style), as well as interest from non-pooling minters.

- Anyone can liquidate

GAS minters whose ETH collateral value falls below the over-collateralization threshold, and get a reward

from delinquent’s ETH collateral.

- A manipulation-resistant moving average

of GAS/ETH price over z

blocks is maintained

as a global variable in the AMM.

- GAS transfer()

is disabled

to incentivize GAS/ETH market depth in the AMM.

- A synthetic instrument that tracks GAS price, as well as futures/options markets of said synthetic, may spawn extra-protocol, on DeFi and/or CeFi, which should further support the listed benefits below.

Expected benefits to the protocol and its stakeholders

1. Risk-minimization

: Ethereum stakeholders have access to credit without custody, oracle, or governance risk. They keep their ETHs, while still being able to pay their GAS costs or pay for other expenses by selling minted GAS for more ETH and/or stablecoins.

1. Capture-resistance

: in case of a contentious fork, the community can fork off knowing they have an uncontrollable credit facility available to them, along with an uncontrollable DEX, Uniswap. These “auto-pilot public financial utilities” are essential in case current custory-, oracle- and/or governance-centralized CFs/stablecoins choose to use their weight to decide the winning fork against community consensus.

1. Added PoS security

: the CF/AMM are sinks for non-staking ETH once ETH2.0 launches. The CF interest rate and AMM trading fees can be made partially proportional to the amount of staked ETH. The extent to which non-staking ETH represents a security threat is debatable, it is however better to have this lever rather than not.

1. Incentive alignment

between the protocol and high-throughput gas consumers (HTGCs) who are long ETH the token and Ethereum the protocol:

- If HTGCs believe GAS/ETH is too low, they go long: deposit ETH \rightarrow

mint GAS. Spend GAS later when it’s expensive.

- If they believe it’s too high they go short: deposit ETH \rightarrow

mint GAS \rightarrow

sell GAS into more ETH. Buy GAS later when it’s cheap.

1. If HTGCs believe GAS/ETH is too low, they go long: deposit ETH \rightarrow

mint GAS. Spend GAS later when it’s expensive.

1. If they believe it’s too high they go short: deposit ETH \rightarrow

mint GAS \rightarrow

sell GAS into more ETH. Buy GAS later when it’s cheap.

1. Predictable capital expenditure

by HTGCs to plan their gas budget: they can “lock-in” the price of their future gas expenditure.

1. Efficient discovery of gas macro-price

: GAS/ETH discovers ~the floor price of a gas unit, i.e. the material cost of 1 gas worth of computation. The aforementioned shorting/longing in (4) contributes to this efficiency. This may also eliminate potential deadweight issues with current JIT gas production/consumption.

Design goals

1. Depth of GAS/ETH market .
2. Minimalism. Fully auto-pilot with no oracle and/or governance components.
3. Backward compatibility. ETH can still be used to JIT pay for gas as it is now.

Contracts

- A) GAS Token

An admin-keyless non-upgradable standard ERC20 contract. Its transfer()

reverts unless _to

is the block producer’s address. Hence, in v1 GAS can only be pooled, or spent on tx fees. This ensures max AMM depth \rightarrow

better discovery of base gas prices. In the distant future, if the depth of GAS market exceeds a certain high threshold,transfer()

can be enabled through a hardfork. Meanwhile, the market can still trade it extra-AMM by proxy through a synthetic that tracks its AMM price.

This does not undermine the enshrined status of ETH as the sole acceptable currency for tx fees because for every GAS spent there is >=1.25x-GAS-worth of ETH locked as collateral

. i.e. GAS is “marked ETH”.

- B) The GAS/ETH AMM Exchange
- zMA_GAS_PRICE

: a global variable representing a volume-weighted moving average of median GAS/ETH (GAS tokens per Wei) price over z

blocks, updated before the first call to trade()

in a given block. Subsequent trades within said block don’t get factored in just yet in zMA_GAS_PRICE

, but they will be the moment trade()

is called again in a future block which inspects []latestTrades

.

- []latestTrades

: an array of GAS/ETH prices during block n

which was the block where the AMM’s trade()

was last called. When trade()

is called for the first time in a block, the zMA_GAS_PRICE

is updated with the previous []latestTrades

, and []latestTrades

is emptied out. The GAS/ETH price of this current trade()

call in this current block, as well as every other trade()

call within the current block, is appended to []latestTrades

, to be consumed in whichever next block trade()

is called, and so on. In brief, trades within block n

get factored-in in the zMA_GAS_PRICE

at block m>n

where block m

is the first time the AMM’s trade()

was called since it was last called in block n

. TODO: All trades within a block are executed as one big aggregate order to thwart any miner shenanigans w.r.t. the ordering the trades (hence latestTrades

becomes a uint

value instead of an array).

- pool()

: deposit ETH and GAS into the AMM to collect trading fees and and interest.

- trade()

: buy GAS with ETH or vice versa, the quote given is according to the curve.

- zMA_GAS_PRICE

: a global variable representing a volume-weighted moving average of median GAS/ETH (GAS tokens per Wei) price over z

blocks, updated before the first call to trade()

in a given block. Subsequent trades within said block don’t get factored in just yet in zMA_GAS_PRICE

, but they will be the moment trade()

is called again in a future block which inspects []latestTrades

.

- []latestTrades

: an array of GAS/ETH prices during block n

which was the block where the AMM's trade()

was last called. When trade()

is called for the first time in a block, the zMA_GAS_PRICE

is updated with the previous []latestTrades

, and []latestTrades

is emptied out. The GAS/ETH price of this current trade()

call in this current block, as well as every other trade()

call within the current block, is appended to []latestTrades

, to be consumed in whichever next block trade()

is called, and so on. In brief, trades within block n

get factored-in in the zMA_GAS_PRICE

at block m>n

where block m

is the first time the AMM's trade()

was called since it was last called in block n

. TODO: All trades within a block are executed as one big aggregate order to thwart any miner shenanigans w.r.t. the ordering the trades (hence latestTrades

becomes a uint

value instead of an array).

- pool()

: deposit ETH and GAS into the AMM to collect trading fees and and interest.

- trade()

: buy GAS with ETH or vice versa, the quote given is according to the curve.

- C) The ETH-collateralized GAS credit facility
- deposit(uint collateralETH)

: deposit collateralETH

Wei's into the facility contract

- mint(uint amount)

: mint amount

GAS tokens where 1.25*amount/collateralETH > zMA_GAS_PRICE

. I.e. collateral maintenance level is 125%.

- liquidate(uint partial)

: aka bite, anyone can deposit GAS tokens, which are burnt, and get 0.1 * collateralETH

reward from delinquent’s ETH collateral (i.e. biter gets a reward equivalent to 10% of ETH collateral). Remaining ETH is sent back to the delinquent. Implementation wise, it is simpler to require liquidators to pay the full amount of debt. However, if we imagine massive debt positions, it might be good idea to allow “partial biting”. In this case, a biter gets 0.1 * partial

(note partial<=amount)

. Because zMA_GAS_PRICE

is a moving average it doesn’t have spikes but rather slowly ascends or descends, and so even in extreme volatility only positions that have already been known for some time (depending on chosen period of MA) to be at risk will be liquidated. This is why a 125% over-collaterlization ratio is sufficient (as opposed to 133%, 150% by Compound and MakerDao, respectively, for their ETH-collaterlization mints/loans).

- withdraw(uint partial)

: anyone who deposit()

ed can withdraw all the collateralETH

collateral (if their GAS debt is 0) or partial

<

collateralETH

of it such that the 125% over-collaterlization ratio maintained: assert this inequality 1.25*amount/(collateralETH-partial) > zMA_GAS_PRICE

- No Oracle

is needed as zMA_GAS_PRICE

is always updated every block the trade()

function is called.

- deposit(uint collateralETH)

: deposit collateralETH

Wei’s into the facility contract

- mint(uint amount)

: mint amount

GAS tokens where 1.25*amount/collateralETH > zMA_GAS_PRICE

. I.e. collateral maintenance level is 125%.

- liquidate(uint partial)

: aka bite, anyone can deposit GAS tokens, which are burnt, and get 0.1 * collateralETH

reward from delinquent’s ETH collateral (i.e. biter gets a reward equivalent to 10% of ETH collateral). Remaining ETH is sent back to the delinquent. Implementation wise, it is simpler to require liquidators to pay the full amount of debt. However, if we imagine massive debt positions, it might be good idea to allow “partial biting”. In this case, a biter gets 0.1 * partial

(note partial<=amount)

. Because zMA_GAS_PRICE

is a moving average it doesn’t have spikes but rather slowly ascends or descends, and so even in extreme volatility only positions that have already been known for some time (depending on chosen period of MA) to be at risk will be liquidated. This is why a 125% over-collaterlization ratio is sufficient (as opposed to 133%, 150% by Compound and MakerDao, respectively, for their ETH-collaterlization mints/loans).

- withdraw(uint partial)

: anyone who deposit()

ed can withdraw all the collateralETH

collateral (if their GAS debt is 0) or partial

<

collateralETH

of it such that the 125% over-collaterlization ratio maintained: assert this inequality 1.25*amount/(collateralETH-partial) > zMA_GAS_PRICE

- No Oracle

is needed as zMA_GAS_PRICE

is always updated every block the trade()

function is called.

Changes to Ethereum Clients

The gas metering functionality in clients is amended to accept GAS as payment for transactions. Backward compatibility is maintained; users choosing to pay with ETH in JIT style as is the case now can do so. If gasprice

0 \rightarrow

the client interprets the tx as v1, and v2 otherwise (v2 = pay from GAS in AMM.balance_of[msg.sender]).

Cryptoeconomics

Interest

Interest on minted GAS is fundamentally a fee that the Ethereum protocol charges minters in return for a valuable service: a guaranteed access to its commodity (block space) at any future moment of time if they keep holding that GAS, or access to immediate liquidity if they short sell it today. The interest formula it TBD

but it factors in:

- A

utilization ratio (amount

/zMA_GAS_PRICE

:collateralETH

ratio).

- B

AMM depth (pool size). The amount of minted GAS that is being pooled.

- C

total ETH staked in ETH2.0, interest rate can be made to inversely track TOTAL_ETH_STAKED

, such that depositing ETH to mint GAS is more attractive the less ETH staked.

Trading fee:

The fees can be determined numerically following, e.g., section 4.1 in this paper.

The GAS/ETH Moving Average (MA)

The higher time frame the MA is (in units of blocks, but those can be mapped to approximate number of hours/days/weeks etc) the more prohibitive it becomes for the GAS/ETH price to be manipulated (for the sake of liquidating minters for example), but the farther away from price it will be (whether above or below). The lower time frame it is (the extreme case being 1 block, at which point the MA is basically just the price itself) the cheaper it is to manipulate the MA but the closer it is to price.

The idea of tracking

the price of an asset as opposed to pegging

to it has and still is being seriously considered for fiat-tracking crypto assets.

There is no “peg” here as GAS/ETH floats, and the integrity of the CF is guaranteed by the safe assumption that ETH has and will continue to have a monetary premium and as such the market will be more than willing to take delinquent’s discounted ETH and bring CF to >= 125% collateral:debt ratio.

The daily MA is potentially the sweet spot between manipulation-resistance and GAS/ETH-price sensitivity, but more analysis of this is needed.

GAS futures and options market:

It is expected that a GAS futures market will pop up. The futures market allow someone buying (selling) GAS today to hedge against an increase (decrease) in GAS prices from now till futures contract’s expiry.

Potential Risks

- The GAS/ETH pool is illiquid: users simply fall back to the current auction mechanism. Since only GAS contributed to the AMM pool accrues interest and fees, minters are incentivized to keep stock GAS pooled and withdraw from the pool JIT style when transacting.
- mitigation

: The tx fees collected by the pool could be made dynamic, inversely proportional to the amount of GAS in the pool.

- mitigation

: The tx fees collected by the pool could be made dynamic, inversely proportional to the amount of GAS in the pool.

- Attacker stocks up on GAS when it’s cheap, launches DDoS on the network when its congested and the GAS price is high*
- mitigation

: (1) block size elasticity (2) There is an incentive during this attack to mint GAS and short sell it, which will bring its price down, thereby dampening the advantage the attacker has (i.e. GAS becomes cheaper, so eventually block space becomes as cheap for everyone else as it is for the attacker), (3) The same can be said about an attacker stocking up on cheap ETH.

- mitigation

: (1) block size elasticity (2) There is an incentive during this attack to mint GAS and short sell it, which will bring its price down, thereby dampening the advantage the attacker has (i.e. GAS becomes cheaper, so eventually block space becomes as cheap for everyone else as it is for the attacker), (3) The same can be said about an attacker stocking up on cheap ETH.

- GAS volatility against fiat: (1) This should not be worse that ETH volatility against fiat since GAS is a less complex an instrument than ETH (assuming it becomes liquid enough) (2) The elastic block size should help counter this.
- mitigation

: transfer()

is disabled. GAS can only be minted, pooled, and spent on tx fees. EIP1559 or equivalent may also help.

- mitigation

: transfer()

is disabled. GAS can only be minted, pooled, and spent on tx fees. EIP1559 or equivalent may also help.

Future work

1. The exact interest and trading fees formulas, and the MA period.
2. Survey of traditional commodity markets esp. price stabilization strategies if there’s something to be learned there.