PhoeniχCoin:
An ERC223 Token Mined Through
Proof-of-Volatility

Norsefire ToCsIcK
March 13th 2018
norsefire_phx@protonmail.com tocsick_phx@protonmail.com

Abstract

This paper introduces the underlying mechanics and interface for PhoeniχCoin (PHX), an Ethereum ERC223-compliant cryptocurrency. The supply of PHX tokens is tied directly to the Ethereum balance of the smart contract underpinning the 'EthPhoenix' decentralised application, and mined using a novel, generalisable technique we refer to as *proof-of-volatility*.

Contents

PhoeniχCoin: An ERC223 Token Mined Through Proof-of-Volatility

Acknowledgements

We both owe a significant debt of thanks to a number of our fellow developers and members of the EthPhoenix community for their support, for their numerous ideas, refinements and proof-reading services. Since the only fair way to do this is alphabetically, we'd like to specifically thank the following:

- **Cardioth** - for his artwork and various comments about content and direction whilst moving house and accidentally flooding his new garage. Enjoy Atlantis!
- **fks** - for his tireless proof-reading efforts and reminding us how punctuation is actually supposed to work. Turns out being a copy-editor actually pays off!
- **oguzhanox** - for being ever-present to bounce ideas off of, general feedback whilst this was in development, and simultaneously driving us mad with random capitalisations of words whenever using Discord from his phone. Seriously, get a new one.
- **Randall** - for proving that Canadians are *exactly* as nice as the stereotype suggests. Your unflagging enthusiasm, feedback and wisdom are probably the reason we all stuck together for this. We owe you a pint of Guinness the next time we're all in Ireland.
- **TechnicalRise** - for being one of the few lawyers we can have an engaging discussion with. In particular, your implementation of the PyrMex smart contract and ideas about the future direction of this project inspired us to finish writing this white-paper and smart contract. This is basically all your fault now.

More generally we would like to thank each and every participating member of the EthPhoenix Discord. You've been an incredible presence in our lives, even if you did drag us back to 4chan. Nonetheless, we (and by proxy, all of the other developers) are deeply grateful for all of your support.

1 Introduction and Context

In this section, we describe the series of events and atmosphere preceding the launch of PhoeniχCoin, its underlying features and idiosyncrasies. We also summarise the most commonly-known cryptocurrency 'mining' techniques used as of the time of this writing. These are proof-of-work (PoW), proof-of-stake (PoS), and proof-of-authority (PoA). We conclude by summarising the novel contributions of this paper.

1.1 The Rise And Fall (And Rise) of Pyramid Contracts

Although the concept of pyramid and Ponzi coins on the Ethereum network has been present in various forms for years,[1] the context of this paper is far more recent. On the 28th of January, 2018, unknown persons registered the domain powhcoin.com, and just before 22:00 UTC on the same day, said persons launched a smart contract.[2] This was "Proof-of-Weak-Hands Coin" (PoWHCoin), describing itself as 'the world's first autonomous and self-sustaining pyramid scheme.'

The core premise was that as additional participants entered the contract, each token's value would grow, with token holders additionally receiving a 10% buy-in fee as a 'dividend' each collected in proportion to held tokens. Eventually, the token sell price would rise high enough for existing incumbents to sell at a profit, thereby reducing the token price for all remaining participants and triggering a crash due to panic sells. Before long, a new price floor would be established (with 'weak hands' shaken out), beginning the cycle anew – providing long-term holders profit through the 'proof of weak hands'.

PoWHCoin was an Ethereum mainnet adaptation of work that first appeared on the Ropsten test network in June 2017.

1.1.1 Ponzi Token

The Ponzi Token smart contract[3] was a proof-of-concept implemented and released by Dr. Jochen Hoenicke (5). The core concepts were:

- Buying Ponzi Tokens would incur a 5% fee, which is distributed proportionally amongst existing token holders (including the new buyer).
- Ponzi Tokens could be sold for 80% of their current buy price in Ether.
- For each Ponzi Token bought, the price and reserve amount would increase (via a custom implementation of the Bancor formula (4)). Similarly, for each Ponzi Token sold, the price and supply would decrease.
- The price and total supply was linked by the equation: $price = \sqrt[4]{supply}$, and the amount in reserve and total supply was linked by $reserve = \frac{4}{5}supply^{\frac{5}{4}}$.

[1] The Ethereum network went live on the 30th of July 2015, with the 'DynamicPyramid' smart contract launching in March 2016.

[2] Etherscan: Contract 0xA7CA36F7273D4d38fc2aEC5A454C497F86728a7A

[3] Etherscan: Contract 0x2CB6ef99FbC78069364144E969a9A6e89E550359

In mid-January, 2018, an anonymous developer encountered the Ponzi Token webpage[4] on Jochen Hoenicke's site. PoWHCoin launched a week later.

1.1.2 Proof-of-Weak-Hands Coin (PoWHCoin) and the Shadow Fork

Some changes had been made to the core concept (the amount of Ether in reserve being lowered to 50% and the buy fee being increased to 10% being the most prominent), but for the most part the smart contract remained identical to the Ponzi Token code on Ropsten. Once the wider online community became aware of this mainnet variant - mostly through Reddit and the 4Chan 'Business and Finance' board, /biz/, the Ether began flooding in.

The initial variant of PoWHCoin was a huge success by any metric for a decentralised application. Over the course of the first forty-eight hours, a total of 3,101.66 Ether via 9,032 transactions were recorded, equivalent to over US$3.7 million using the average Ethereum closing price over the 28-29th January).

Less than forty-eight hours after the launch of PoWHCoin, an unreachable page was committed to the Git repository of the front-end, referencing an upcoming variant of PoWHCoin that promised additional dividends to those who participated. Given the mania of the previous two days, this was eagerly picked up on and anticipated by those already 'invested' in PoWHCoin. Near midnight (UTC) on the 31st of January, the 'Shadow Fork' smart contract[5] was deployed.

The Shadow fork implemented an additional fee of 10% on all sales of tokens purchased through its smart contract, and rapidly collected over 867 Ether (just short of a million US dollars at the time) in the course of two hours. Unfortunately, the implementation of the sale fee was such that the contract was susceptible to a uint underflow, causing the contract to enter an exception state and temporarily 'lock up' participants' Ether. A combination of this and the disclosure of an exploit[6] in both variants ultimately led to the theft[7] [8] of roughly 1,600 Ether by an unknown perpetrator. We will not go into the details of these exploits here, as they have been covered in depth elsewhere online, particularly on Reddit[9].

1.1.3 EthPyramid and EthPhoenix

In the aftermath of the Shadow contract's failure, a separate group of programmers[10] (the 'EthPhoenix developers') took it upon themselves to see if the underlying smart contract of the Shadow fork could be made exploit-proof. Whilst it has since been seen to be

[4] The Ponzi Token testnet contract was self-destructed on the 1st of February, 2018.

[5] Etherscan: Contract 0x9f4fD6c336388F2ab7DC7bBe4740aE7B88B880D7

[6] Etherscan: Transaction 0xb08fb4ec0b3c7ed15579fa65c84778296f858d48e51b86e140f5ce5350ce029f

[7] PoWHCoin drain: 0x496c0411f52978dfd7953b7e6965465977162bfaf7b88c0c78fcdc97cd395d62

[8] Shadow Fork drain: 0xf8c77b539211bc3bc4abb1779d753535195fa3868456bf22a921f419e5d34e37

[9] /r/PoWHCoin: Comment in "PoWHCoin hacked, 866 ETH stolen. The "mathematically-sound Ponzi scheme" smart contract has all funds stolen due to code exploit." by /u/longtermsec

[10] Disclosure: the author of this paper is a member of this group.

possible to do so whilst maintaining ERC20 compatibility[11], it was decided at the time to remove all inter-wallet transfer functionality, since in practice the only point of contact that is *necessary* is the smart contract itself. This modified smart contract[12] was released as ‘EthPyramid’ (EPY) at 05:00 AM UTC on the 2nd of February, 2018.

The above smart contract has been running continuously since launch, with a separate front-end pointing at the same contract address under the ‘EthPhoenix’ name[13] (EPX) added on the 23rd of February, 2018. However, as mentioned above, the smart contract’s tokens are not ERC20 compliant and cannot be traded. Whilst this meets the specifications of the initial vision of the project, it has proven limiting insofar as adding layered functionality for the community is concerned.

At this point, we are familiar with the context within which this work is presented. We now take some time to discuss the most common existing techniques for ‘mining’ cryptocurrencies that are currently in use.

1.2 Mining: Techniques and Limitations

As one of the core contributions of this paper is the introduction of a new technique for generating tokens, we briefly cover the techniques which most cryptocurrencies use. Fundamentally, these techniques serve to add value to the cryptocurrencies they produce due to the costs involved in executing them (i.e. hardware, bandwidth and electricity), whilst simultaneously acting as a confirmation/validation protocol for transactions made across their respective networks.

1.2.1 Proof Of Work

Proof-of-work (PoW) is by far the most commonly-used technique for adding new blocks to a cryptocurrencies’ blockchain, involving the expenditure of a non-trivial amount of computational power with the aim of solving a problem such as a partial hash inversion or a guided puzzle tour. PoW acts as an economic measure (in that there is an economic cost to participating in the process) designed to ensure a consistent view of the distributed ledger which comprises the blockchain. More importantly, it enforces a delay between the creation of new blocks, thus forcing the ‘serialisation’ of transactions and thereby preventing double-spend attacks.

Given that the sole purpose of PoW is to act as a bottleneck for transaction validation, the problem that a PoW-based cryptocurrency poses for its miners can in theory be *anything* of sufficient computational complexity, under the proviso that the *solution* to said problem has both a deterministic relation to the contents of a block and can be verified in a short amount of time. For example, Bitcoin’s SHA-256 PoW algorithm is a variant on the computation

[11] Etherscan: Contract 0xB3775fB83F7D12A36E0475aBdD1FCA35c091efBe

[12] Etherscan: Contract 0x2Fa0ac498D01632f959D3C18E38f4390B005e200

[13] It turns out that having the word ‘Pyramid’ in a front-end brings negative connotations to mind!

of the pre-image of a SHA-256 hash function (8), and the Ethereum Dagger-Hashimoto PoW algorithm (now evolved into a form commonly referred to as Ethash) is a memory-hard computation on directed acyclic graphs (DAGs) with an I/O bound to increase the algorithm's resistance to ASIC machines (3).

The main downsides to using PoW are twofold: ecological impact and the need to consider post-quantum cryptography. As of the time of this writing, the amount of electricity expended daily in the endeavour of Bitcoin mining is on par with that of Israel (1), and developers are considering moving away from PoW validation schemas as a result. In addition, given that the vast majority of current PoW schemas are fundamentally based in public-key cryptography, they can (hypothetically) be easily broken by a quantum computer of sufficient power using Shor's algorithm for prime number factorisation (11). Whilst there *are* hash-based algorithms currently in existence which are believed to be quantum-proof (such as those using the Merkle signature scheme (7)), a sizeable number of cryptocurrencies are likely to be unwilling/unable to alter their validation protocols in time, if ever.

1.2.2 Proof Of Stake

One less power-intensive alternative to proof-of-work while ensuring a distributed consensus is proof-of-stake (PoS). A PoS algorithm can factor in the size or 'age' of a wallet when determining who has the right to forge a block and receive the associated reward, often including a degree of random selection to prevent the largest or oldest holder from having unanimous control over the ledger.

One particular example of a coin that relies entirely on PoS and has done since inception is Nxtcoin (NXT). NXT decides on the next 'generator' of a block as follows: each account participating in the mining process determines its' unique target 'hit value' via a formula parameterised by the account balance, and then calculates a hash value produced by signing the generation signature of the *previous block* with it's public key (9). The first account to produce a hash value lower than its target hit value wins the right to forge the next block, with the probability of doing so being roughly equivalent to the proportion of staked coins it holds in comparison to all staked coins currently involved in mining (10).

Using a PoS schema eliminates the advantage which ASIC miners have in a PoW environment, as the capability to mine coins is linked predominantly to the *existing* economic interest a participant has in the network. Furthermore, in addition to the lower energy requirements of PoS, some schemas can implement a disincentive for attempting to participate in the forging of fraudulent blocks (a 'nothing-at-stake' attack) by seizing the stake of any account that demonstrates malicious behaviour such as extended participation in the mining of a fork in the blockchain.

Ethereum has proposed precisely such an algorithm for a PoS approach to mining in the past (Slasher). However, this has now been dropped in favour of the Casper algorithm, due to be implemented as part of the Constantinople hard-fork.

1.2.3 Proof Of Authority

As mentioned in the previous section, cryptocurrencies using PoS without a disincentive can find themselves subject to nothing-at-stake attacks, where malicious actors find it profitable to mine multiple forks of a blockchain in addition to the longest - and hence most canonical - sequence. One way to mitigate this is to use proof-of-authority, a technique in which the trustworthiness in the identity of a generator forms a part of their 'stake'.

Accounts that are holders of a PoA cryptocurrency can choose to delegate their voting power to either themselves or a separate account, and such designated 'authorities' are granted the right to forge blocks, often in exchange for a share in the profits running an authority node confers. Due to this profit incentive, delegates are expected to both undertake the responsibility of maintaining a full uninterrupted node while keeping true to their word. Failure to do this often results in votes (and implicit trust) to be reassigned to other, more reliable delegate authorities.

This technique is conceptually similar to *delegated proof-of-stake* (DPoS). One cryptocurrency which makes use of the latter is Ark - the top 51 accounts (as measured by voting balance) become *delegates*. Delegates can forge 422 ARK tokens daily,[14] on top of any transaction fees in the blocks they forge, and different delegates offer various percentages of the profits to their voters in exchange. The primary criticism levelled against PoA is that whilst it works well in a private blockchain setting, in a wider sense the centralisation of power amongst a select few reintroduces precisely the same issue which cryptocurrency initially set out to solve.

There are a myriad of other cryptocurrency 'mining' techniques, such as the proof-of-capacity (aka proof-of-space) approach most widely known via Burstcoin (2). For our purposes, however, the three techniques that we have identified thus far are a sufficient context for contrast against proof-of-volatility.

1.3 Contributions

Whilst the underlying smart contract began as an exercise in academic curiosity (and should *not* be considered a financial investment), this paper establishes the framework for what the author believes are genuine novelties within the Ethereum research space. In particular, they are -

- A presentation of a model on which queryable values from smart contracts can be used for generating Ethereum tokens (proof-of-involvement, see §3.1).
- A specific instance of proof-of-involvement describing an algorithm which dictates proportional token generation according to the stake held by a participant in a separate smart contract (proof-of-volatility, see §3.2).

[14] This daily ARK forging figure is true as of March 2018, but may change as time goes on.

- A smart-contract detailing an ERC223-compliant Ethereum token active on the main network which can be mined through a hitherto unused technique utilising an existing smart contract as the underlying generator (PhoeniχCoin, see §4).

We now move on to a discussion about the implementation of Ethereum smart contracts as underlyings, focusing specifically on the features of the EPX contract which permit us to use it as an underlying for PhoeniχCoin.

2 Underlyings as Smart Contracts

In this section, we will discuss the (theoretical) rationale for representing physical commodities in a derivatives smart contract for the purposes of hedging and trading. We will also discuss the ways in which the typical Greek risk measures could be measured for such contracts, before moving on to discussing a real-world example. We then describe the features of the EPX contract which make it amenable for use as an underlying on which we can peg the generation of PHX coins.

2.1 Modelling Commodities

Nearly all seminal texts on hedging theory begin by describing a farmer who wishes to protect their earnings by setting the price of the wheat they grow in advance. By doing this, they partially inoculate themselves against the whims of an uncertain supply and demand curve: they will not enjoy the additional profits that would be associated with a year in which wheat is in high demand, but similarly will not fall victim to potential losses associated with a year in which the price of wheat falls dramatically.

This idea in its various forms is fundamental to the existence of most derivative contracts in the financial markets, such as futures, forwards and swaps. In homage to academic tradition, we will continue with the wheat analogy, and imagine that after a farmer has sold futures on their crop to a trader at an exchange, that trader decides to list these futures through the Ethereum blockchain in the form of a token called WheatCoin.

The WheatCoin smart contract is released entirely pre-mined (i.e., no more WheatCoin can be created), with a total supply of 30,000. The price is initially pegged at US$5.00 per coin (with each WheatCoin corresponding to one bushel of wheat), and includes a `buy` function which is time-locked for six months, at which point it opens for one day, and allows holders to redeem their tokens at a price (in Ether) decided by the spot rate for wheat on that day. Since the future price is yet to be determined, the WheatCoin contract initialisation is accompanied by US$150,000 worth of Ether to cover the buy-back cost at the time of sale.[15]

[15] This isn't quite how futures contracts work in practice - in the real world of commodities futures, no money exchanges hands until the settlement date. Nonetheless, this is the protocol followed by Ethereum smart contracts - creators must be capable of buying back all of the tokens they issue.

The WheatCoin token is then purchased by third parties (adding another US$150,000 in Ether to the contract balance in the case that all tokens are bought), and may possibly be traded around through decentralised exchanges. Ultimately, the only 'true' counterparty to the token is the contract itself. Moreover, the token has a pre-defined lifespan before becoming worthless: precisely six months and a day from its creation. The trader who created the contract sells these tokens in the hope that the price of wheat will drop (and the price of Ether will stay stable or increase) by the time the redemption window opens. This will allow the `buy` function to be for a lesser amount than initially pegged to at creation - and it is this amount that holders of WheatCoin will be *forced* to accept, as the alternative is zero the next day - leaving a net amount of Ether in the contract as profit.[16]

In this way, we can expose commodities typically traded in bulk via classic exchanges to the cryptocurrency market, and allow investors in the space to participate in the commodities derivatives market. This, of course, relies heavily on the trustworthiness of a contract to set the buy-back price to an honest value (and is therefore perhaps best left to an external source as an event to be triggered at a given date), but could prove interesting in the future.

2.2 Abstracting Away: Using Financial Greeks

No discussion about the value of derivative contracts is complete without a concomitant section on 'Greeks'. In short, these are measures of *risk* associated with an underlying, and frequently used by derivatives traders in constructing strategies for minimising the impact of a downward turn in market conditions. Each Greek corresponds to the sensitivity of the price of a derivative in relation to the change of a parameter of the underlying itself.

The key Greeks that are of interest are the first-order derivatives such as *delta*, *theta* and *rho*. There are a significant number of other Greeks corresponding to higher-order derivatives of the value function, but for our purposes the 'entry-level' Greeks suffice.

Imagine that a smart contract exists which offers European-style options[17] on an ERC20 token XYZ in the following way:

- We assume that the contract only allows for options to be purchased for a single unit of the underlying. In practice, options are usually bundled into 'lots' of 100 units, but our example easily generalises to arbitrary amounts.
- The contract offers two 'flavours' of tokens corresponding to call and put options. The prices for these are changed at pre-defined intervals via an external oracle, and historical prices are added to an array which the contract exposes in its ABI.

[16] This is slightly oversimplifying the issue, as nothing is compelling the trader to meet the margin call if the price of wheat *increases* and the contract cannot honour its buy-back obligation. Let us assume they are an honest party, and will deposit any additional Ether required at the time of buy-back price determination.

[17] An option grants the buyer the right to buy or sell an underlying at a future date, at a rate determined at the time of purchase. A European option can only be exercised on the maturity date, in contrast to American options which can be exercised at any point during their lifespan.

- A single call option (right to buy) can be purchased from the contract by depositing an amount of Ether equal to the current price of the option and the pre-defined strike-price value of the underlying. The latter amount is placed in a multi-signature escrow account with the smart contract and buyer as co-signatories. The contract then places one token of XYZ into the same account, and issues a secret calculated by signing the public key of the escrow account with the public key of the contract.
- At the maturity date, the buyer can exercise their option by showing the smart contract the associated secret - in which case the assets in escrow switch ownership (see below) - or either indicate rejection of the option or fail to act in time, in which case the assets are returned as appropriate.
- A single put option (right to sell) can be purchased from the contract in the same way, but with the assets that each counter-party places in escrow reversed.
- Call and put tokens can be freely traded on the Ethereum network as WheatCoin tokens are, provided that there is a method for securely transferring secrets between third parties. One consideration to bear in mind, however, is that there is one token of the underlying in escrow to be bought (or sold) by the bearer of the option. As such, both the current market price and strike-price of the underlying will be priced in to the exchange of option tokens on a secondary exchange.

Given the above, at any given point we have access to full current and historical information about the price of both the options and the underlying itself, and we also know how much time exists before maturity. With these, it is possible to expose a function `delta`[18] corresponding to the first-derivative of the price V of an option with respect to the price S of its underlying, or more formally, $\Delta = \frac{\delta V}{\delta S} \simeq \frac{V(S+\delta S)-V(S)}{\delta S}$. Similarly, with access to the historical prices of the underlying, other functions such as `theta` which represent the sensitivity of an option's price to the volatility of its underlying ($\Theta = \frac{\delta V}{\delta \sigma}$) can be exposed.

Unfortunately, no widely used language for building smart contracts currently has native support for floating point arithmetic, and so attempting to calculate anything other than an approximation of a definite integral using the `int` datatype[19] is bound to end badly. This means that the computation of any Greek would have to take place off-chain and be reported back as an `int`, resulting in a gas-costly, time-consuming operation. Nonetheless, in theory it is possible to do this, and the author suspects that as cryptocurrencies become increasingly entwined with traditional markets and ever more exotic derivatives become widely available, solutions will be proposed and incorporated.

[18] The *absolute value* of the delta of an option is a number between 0 and 1, representing the degree to which a change in the price of the underlying impacts the price of the option. An alternate, more intuitive view is that the delta corresponds to a multiplier for the 'amount' of the underlying that the option is currently worth. An option that is guaranteed to never be exercised (is significantly 'out-of-the-money') has a delta of 0, and vice versa.

[19] The delta of a put option is always negative.

2.3 "Real World" Example: The Venezuelan Petro

Although the examples we have presented throughout this section are purely theoretical,[20] there is one recent, particularly relevant example in the real world: the Venezuelan *Petromoneda*, or 'Petro'. On the 3rd of December, 2017, Venezuelan President Nicolas Maduro announced the Petro in a live television address, stating that it would be backed by the country's (considerable) natural resources, predominantly oil.

The problems facing the Petro are legion; the Venezuelan National Assembly considers it an illegal debt issuance, the United States believes that investing in the Petro circumvents sanctions as an attempt to access international funding, the Petro can only be purchased in US dollars but (at present) can only be used to pay Venezuelan taxes priced in bolívar, Petros cannot be redeemed directly for their underlying, and their value is determined solely through a 'governmental valuation' of their oil and mineral prices rather than any globally accepted oil-price standards. The white paper for the project is available,[21] but has seen multiple changes on a regular basis even since the commencement of their pre-sale.

Despite this being the disappointing fore-runner of commodity-backed and governmentally-issued cryptocurrencies, other countries have been quick to take notice and attempt to mimic the model. Cambodia intends to propose a national cryptocurrency named 'Entapay' at an international summit in Phnom Penh in March 2018, and (encouragingly), the Marshall Islands recently became the first country in the world to announce plans to issue it's own cryptocurrency - the Sovereign (SOV) - which is accepted as legal tender in conjunction with the US dollar.

2.4 Our Example: EthPhoenix

This brings us, finally, back to the EthPhoenix (EPX) smart contract earlier described in §1.1.3. As we have described, the nature of the EPX contract is that all purchases and sales made through the contract are subject to a 10% fee which is distributed to all participants regardless of whether they joined the contract before or after the user making the transaction. The primary factor in accumulating these 'dividends' is therefore market movement; either in the hopes that enough tokens are purchased after entry to produce a profit at the point of sale (recalling that each token purchased increases the buy and sell prices by a fraction of a percent and vice versa), or by holding steadfast until sufficient dividends have been accumulated to offset any losses incurred by the buy and sell fees.

Over the course of the contract's lifetime, however, members of the community noticed that there were days where there was almost no motion within the contract, to the point of stagnation. As such, an external programmer produced a second-layer smart contract which interfaces with EPX in order to stimulate activity.

[20] And should probably remain so for the time being, as various securities authorities worldwide would likely take a very dim view of unregulated commodity derivatives trading!

[21] Petro Whitepaper [ESPAÑOL]: last accessed 13th of March, 2018

2.4.1 The PyrMex Futures Smart Contract

The PyrMex contract is, at it's core, a guessing game. Whilst it describes itself as providing 'futures' on the EPX contract balance, it can be better summed up as a betting game with encouraged manipulation. It operates as follows:

- Every day, the smart contract seeds six 'buckets' where bettors anticipate the total balance of the EPX contract at a settlement time of 05.00 GMT.
- The 'multiplier' values associated with all buckets are identical to begin with, but are dynamically adjusted as bettors stake their positions - more bets into a particular pot will simultaneously lower the multiplier for that pot whilst increasing the others.
- There is a progressive vigorish imposed on bettors as the day goes on: three hours of free wagering, followed by a 5% house take, then increasing to 10% and finally 14.3%. This serves to incentivise earlier betting, and to impose a 'tax' on the market information for participants making last minute bets.
- The bets close at 04.00 GMT, one hour prior to settlement. Any transactions which are processed after this time are refunded to the bettor, after factoring in gas.
- In the hour between bets closing and contract settlement, manipulation of the EPX balance takes place, with various bettors or current holders either withdrawing or depositing Ether into the EPX contract to bring the balance within the scope of the bucket they have staked a bet upon. Other participants will be doing the same, either as part of a group, or individually. Essentially, the contract acts as normal but with the added impetus of an imminent contract settlement.
- Importantly for participants in the EPX contract, this external motion benefits them in the form of dividends distributed amongst all holders, regardless of whether they are participating in the PyrMex game or not. Similarly, those participating in PyrMex are not necessarily participants in EPX: they may be either betting on a bearish bucket because of negative sentiment, or simply following a larger bet that they observed being placed earlier.
- Once the settlement time is reached, the balance is locked in, and all bettors within the winning bucket receive the full contents of the pot at the multiplier that was locked in when bets closed. Payments are then distributed for winners to withdraw, and the next round begins after an hour cool-down period.

These price movements in the EPX contract take place whether its holders choose to participate in PyrMex or not, and whilst the buy and sell price may shift in a way that is undesirable to them, their willingness to stand firm is rewarded by dividends.

We started this section talking about modelling underlyings and the various types of financial Greeks which could theoretically be implemented as a result. We mention PyrMex because it provides a regular and external source of *volatility*. It is this volatility that we make use of, and we explain how in the remainder of this paper.

3 Proof of Involvement

The model upon which the EthPhoenix smart contract operates is currently susceptible to two particular criticisms that have not been taken lightly. The first is that when first launched, days went by without much movement in the contract, leading to the impression of stagnation. The second is that the token itself is not tradeable on the Ethereum network due to non-ERC20 compliance, as the facilities to do so were explicitly removed from EPX prior to launch.

To paraphrase slightly, one of the key concepts behind Koster and Wright's book 'A Theory of Fun for Game Design' (6) is that an online community should be rewarded for their continued loyalty; be it through giveaways, achievements, or providing means of engagement with their fellow players that are related in a sense to their common bond. That is to say, we wish to provide a mechanism for rewarding participants for their involvement in EPX in a way that doesn't simply involve more buying and selling from the contract.

3.1 A General Model

The first problem to address here is that of EPX's non-ERC20 compliance. With no ability to transfer tokens amongst players, and a 10% buy and sell fee on any transactions with the smart contract in any event, it would appear that such an underlying smart contract does not lend itself to wider engagement (and indeed, this was part of the original design).

However, as we have described in §2.2, there are instances in which we can compute values with informative value – such as the Greeks – based on nothing more than knowing the historical values of certain parameters, such as buy prices and recorded balances. We posit that any contract which allows the querying of a non-negative, rational and bounded number can serve as the underlying upon which tokens for an auxiliary cryptocurrency can be minted, provided that there exists a way for the minted tokens to be distributed in a manner proportional to the amount of commitment a contract participant demonstrates. We call this concept *proof-of-involvement*.

3.2 A Specific Example: Proof of Volatility

The EthPhoenix smart contract exposes a number of functions returning information about the current state of the contract, such as `totalSupply` (a dynamic number, since tokens which are sold are burned) and `dividends` (as accumulated by a particular address). Since these values can be queried by other smart contracts themselves, we can construct a smart contract α that derives the absolute value of daily volatility in EthPhoenix as follows:

1. α sends a small amount of Ether into EPX, and records both the amount of tokens and the dividends (D_0) received from it's initial purchase.

2. When queried at a future point T, α queries its updated dividend balance (D_T), as well as the current total supply of EPX tokens.

3. Dividing the number of tokens that α holds by the current total EPX supply produces the proportion of tokens ($EPX_{\alpha T}$) which α holds.

4. The total amount of dividends (D_{Ω}) paid out between time 0 and time T to all participants can then be calculated as $D_{\Omega} = \frac{D_T - D_0}{EPX_{\alpha T}}$.

Henceforth, we refer to the above as the *proof-of-volatility* (PoV) algorithm. This is a specific case, however, and the concept of using figures derived from external smart contracts can be similarly leveraged, and it is this type of algorithm that we refer to as the skeleton of the proof-of-involvement technique.

The question that we have left unanswered up to this point, however, is how we can use PoV in a meaningful way. To this end, in the next section, we introduce PhoeniχCoin, and describe how PoV is used to mine it.

4 Introducing PhoeniχCoin

PhoeniχCoin is designed to act predominantly as a gaming utility token, with a number of decentralised applications currently in development as of the time of this writing which are out of the scope of this paper. Nonetheless, the introduction of a tradeable token which stands independently to - and yet is inextricably linked with - the EthPhoenix smart contract is a way for the EPX developer team to thank the community for their faith and support. Moreover, a tradeable token provides far more flexibility in terms of the ecosystem which can be built around it.

4.1 Properties and Mining

PhoeniχCoin is fundamentally quite simple. It has been released with a pre-mine of 100 tokens for the purpose of initial giveaways, has no maximum supply, and is defined up to 18 decimal places (as is the case with most Ethereum tokens).

In order to mine PHX, an EPX participant must follow the below steps:

1. Visit the PhoeniχCoin page at `https://phxcoin.io`.[22]

2. Clicking “Begin Mining”[23] initiates a `mine()` transaction to the contract. If it is the first time an account β is mining, three data points are recorded: the block timestamp δ in which the transaction was processed, the EPX dividend balance currently held by the PHX contract D_{δ} and β’s current EPX balance EPX_{δ}.

[22] You will need MetaMask installed, available at https://metamask.io

[23] This is currently the case with the existing front-end, but is likely to change in future. Alternatively, mining can be initiated by sending a transaction to the contract with zero Ether and `0x99f4b251` as an additional data parameter.

3. After a *minimum* of 24 hours has passed, an account can mine again. Using the PoV algorithm described in §3.2, PHX calculates the *total* number of EPX dividends D_τ that have accumulated during the elapsed time-period τ.

4. The PHX contract then determines β's *current* proportion of the total EPX token supply $EPX_{\beta\tau}$, and gives β PHX coins equivalent to $\frac{D_T}{EPX_{\beta T}} \times \frac{\tau}{\tau-\delta} \times \lambda$, where λ is a factor allowing for 100 PHX tokens to be mined for every Ether produced in EPX dividends.

5. The above step has a requirement that β's token balance $EPX_{\beta\tau}$ at the beginning of a subsequent mine (when previously mined PHX tokens are distributed) is greater than zero. The net effect of this is that if β begins to mine and then sells their EPX tokens during that mining phase, they do not receive any PHX tokens for that phase.

6. The divisor $(\tau - \delta)$ acts as a time-decay modifier, making it most profitable to mine exactly every 24 hours. Moreover, the algorithm is designed such that if β does not mine for seven days or more, their balance is reset, they receive no PHX tokens for their current mine and are treated as if they are once again mining for the first time.

4.1.1 ERC223 Compatibility

The generation of PHX tokens through EPX volatility can be viewed in a similar way to the way in which a token such as NEO generates GAS as a utility coin to launch smart contracts on its platform. However, PHX coins serve a slightly different purpose, as they are intended to act as tokens for a series of games built around a common base (namely the Phoenix theme, for the EthPhoenix community). Provided that they are listed on decentralised exchanges – such as IDEX and ForkDelta – it is possible to imagine that PHX coins will eventually themselves have an Ether value as decided by market forces.

To that end, we have made PHX an ERC223-compatible token. ERC223 is a token standard initially proposed in March 2017 which consolidates the `Transfer` and `TransferFrom` ERC20 functions into a single `Transfer` function accepting three arguments,[24] namely a target address, a value (for Ether) and a data payload. ERC223 contracts which *accept* tokens must implement a `tokenFallBack` function which decides how to handle each type of token which could be sent to it.[25]

The reason for doing this is that users commonly confuse the two transfer functions when sending tokens to either a wallet or a contract, and as a result, tokens can be – and frequently are – locked up in contracts with no way to retrieve them. ERC223 aims to solve this by reverting transactions to receiver contracts which do not provide a `tokenFallBack` function, but allowing wallet receivers to process tokens as is the case with ERC20.

[24] The original `Transfer` function only accepted two arguments.
[25] ERC223 tokens are backwards-compatible with ERC20 tokens.

We have chosen to implement PHX as an ERC223 token in an attempt to future-proof the coin: we envisage a shift of Ethereum platform tokens to the ERC223 protocol sooner rather than later, and a multitude of auxiliary contracts interfacing with the PHX contract in various ways.

4.2 Contract Address and Interface Details

This section concludes the paper by providing function hash table for the PhoeniχCoin smart contract. Please note that the contract source code will remain closed until the 1st of June, 2018, at which point it will be verified on Etherscan.

4.2.1 PhoeniχCoin Contract Address

The PhoeniχCoin address is `0x16Ea0DC095ed8c32eFf23BF0d5F31CdFC50A10Fd`, and was created in Ethereum block 5243778 on the 12th of March by transaction `0x1141dd6c59953ca450861cb5aa0cf28d698800a36e65785e9a11e3fd45373ff1`. There are also two ENS (Ethereum Name Service) addresses which resolve to the contract, namely `phxcoin.eth` and `phoenix-coin.eth`.

4.2.2 PhoeniχCoin Function Hash Table

```
{
  ”dd62ed3e”: ”allowance(address,address)”,
  ”095ea7b3”: ”approve(address,uint256)”,
  ”70a08231”: ”balanceOf(address)”,
  ”e171b847”: ”canMine()”,
  ”313ce567”: ”decimals()”,
  ”9d118770”: ”destroy(uint256)”,
  ”99f4b251”: ”mine()”,
  ”c33253b8”: ”minedAmount()”,
  ”38e38a80”: ”miningCooldown()”,
  ”06fdde03”: ”name()”,
  ”c65036fb”: ”newMiner()”,
  ”95d89b41”: ”symbol()”,
  ”18160ddd”: ”totalSupply()”,
  ”a9059cbb”: ”transfer(address,uint256)”,
  ”be45fd62”: ”transfer(address,uint256,bytes)”,
  ”23b872dd”: ”transferFrom(address,address,uint256)”
}
```

References

(1) Digiconomist: Bitcoin Energy Consumption Index - https://digiconomist.net/bitcoin-energy-consumption - Last accessed: 13th March 2018

(2) Dziembowski, S., Faust, S., Kolmogorov, V. and Pietrzak, K. (2013) Proofs of Space

(3) Ethereum GitHub Repository - Dagger-Hashimoto Page - https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto - Last accessed: 13th March 2018

(4) Hertzog, E., Benartzi, Guy and Benartzi, Galia (2018) Bancor Protocol: Continuous Liquidity for Cryptographic Tokens through their Smart Contracts

(5) Hoenicke, J. (2017) Ponzi Token A Self-Trading Token On The Ethereum Network - https://test.jochen-hoenicke.de/eth/ponzitoken/ - Last accessed: 13th March 2018

(6) Koster, R. and Wright, W. (2004) A Theory of Fun for Game Design

(7) Merkle, R. C. (1988) A Digital Signature Based On A Conventional Encryption Function

(8) Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System

(9) Nxt Community (2014) Nxt Whitepaper

(10) Popov, S. (2014) A Probabilistic Analysis of the Nxt Forging Algorithm

(11) Shor, P. W. (1994) Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer