{Set} Protocol: A Specification for Token Abstraction

By: Felix Feng

Version 1.0

November 14, 2017

Abstract

We describe a specification for a new primitive that facilitates the low-cost, trustless creation, and exchange of a {Set}, a collateralized basket of ERC20 tokens on the Ethereum Blockchain. {Set}s serve as an abstraction for end users who want to think about higher-level token concepts without dealing with the details of specific tokens. {Set}s are a superset of the ERC20 token standard with issue and redeem functionality, allowing for the atomic swap of the {Set}s and their underlying tokens. As an investment, {Set}s are similar to index funds (e.g. S&P 500, DJIA) and exchange traded funds (ETFs) in traditional financial services, allowing users to easily get exposure to a multitude of tokens. As {Set}s are ERC20 tokens and composable, it is possible for one token to represent a limitless number of other tokens.

Contents

1 Introduction **3**
- 1.1 The Tokenization of the World . . . 3
- 1.2 The Pains of Token Complexity . . . 4
- 1.3 Abstraction as a Tool . . . 5

2 {Set} **5**
- 2.1 Overview . . . 5
- 2.2 Properties . . . 5
- 2.3 Benefits . . . 6

3 Use Cases **7**
- 3.1 Index Token / ETF . . . 7
- 3.2 Processing Files . . . 9
- 3.3 Health Care Package . . . 9
- 3.4 Other Use Cases . . . 10
- 3.5 Issuance Demonstration . . . 10

4 Smart Contract **12**
- 4.1 Underwriting (constructor) . . . 12
- 4.2 Token Issuance (issue) . . . 13
- 4.3 Token Redemption (redeem) . . . 14

5 Future Infrastructure **15**
- 5.1 Registeries . . . 15
- 5.2 Javascript Library . . . 16

6 Existing Work **16**
- 6.1 Individually Managed Index Tokens . . . 16
- 6.2 Prism . . . 16
- 6.3 Bancor . . . 17
- 6.4 0x Protocol . . . 17

7 Ongoing Research **18**
- 7.1 Summary . . . 18
- 7.2 Acknowledgements . . . 19

8 References **19**

9 Appendix **21**

1 Introduction

1.1 The Tokenization of the World

With the creation of the Bitcoin blockchain and subsequently the Ethereum blockchain, anyone can create their own digitized token to represent securities, goods, services, real world assets, etc. As of November 2017, there are 1,300 tokens listed on CoinMarketCap that represent over $200 billion in value and a long-tail of obscure tokens not tracked by CoinMarketCap [1]. Tokens are being created for numerous purposes and representations including:

- **Currency on Blockchains / DAGs**: Blockchains (e.g. Bitcoin, Ethereum, NEO, Zcash), directed acyclic graphs (e.g. IOTA, HashGraph), and their forks (e.g. BitcoinCash, Ethereum Classic) have their own representative tokens that are used to incentivize parties to process transactions.

- **Protocols**: Protocols such as 0x, Kyber Network, Polkadot, and Raiden are issuing utility tokens that are used to transact on their networks and support decentralized governance.

- **Digital Goods**: Gaming communities, streaming services, content platforms, and gambling sites are using tokens to represent virtual goods, incentivize direct payment of content creators, and create economic systems that bypass traditional intermediaries.

- **Derivatives**: Projects such as dy/dx and Dharma are allowing anybody to issue options, short sells, and loans that are represented by tokens. Projects such as MakerDAO, Basecoin, and Fragments are creating tokens that aim to resist fluctuations in value.

- **Real World Assets**: Projects such as TrustToken and DigixDAO are bringing the tokenization of real-world assets.

- **Securities**: Blockchain Capital, a cryptocurrency hedge fund, has issued their own securities to represent stakes in their fund. Polymath is building a platform to tokenize financial securities. Numerai is using a token to reward data scientists for generating fund profits.

1.2 The Pains of Token Complexity

There is growing complexity and pains in dealing with an ever-growing number of tokens. This is due to the lack of a standard for thinking about and dealing with a multitude of tokens. Some of these pains are detailed below:

- **Transaction Costs**: Transaction fees and the effort for sending numerous tokens can be burdensome. External accounts (standard address in Ethereum) need to construct, sign, broadcast, and pay a transaction fee for each token transfer. The transaction fees and effort of dealing with a multitude of tokens grow linearly with the number of tokens transacted. If one wants to send 100 different tokens using a standard Ethereum external account, one would need to sign and broadcast 100 transactions and thus pay 100 transaction fees. This is not scalable and is a subpar user experience.

- **Cognitive Overload**: As the number of tokens that users need to interact with grows, the cognitive load required for management of tokens grows as well. For each token they own, users must understand each tokens qualities including price, monetary policy, utility, liquidity, interface, etc. Each token needs to be treated individually, and users must retain everything in their head at once.

- **Client Limitations**: Ethereum clients such as Parity, Mist, and Metamask are limited in their ability to manage groups of tokens. Clients dont have features that allow the grouping together of tokens or batch transfers. Thus, common operations like viewing, transferring, and calling functions on multiple tokens are processes that users need to manually and iteratively perform.

- **Investor Pains**: Investors who hold a multitude of tokens lack the tools to effectively acquire, manage, and analyze their investments. Standard active portfolio management practices such as rebalancing, bucketing, and transfers become onerous and repetitive processes. Analyzing groups of tokens requires individually aggregating the prices of tokens and bucketing tokens must be done manually.

  The exchange experience is also painful. Transferring tokens from exchanges requires manual, 1-token-at-a-time withdrawals. Trading fees on exchanges add up when purchasing a multitude of tokens a single token at a time.

- **Developers Pains**: Developers lack the paradigms to hide the details of multiple tokens from users on the protocol level. Information hiding must be done

on the application-layer of the stack vs. natively in the protocol layers. Developers also need to understand all the tokens that their system deals with. Finally, it is difficult to create a good user experience to onboard new users if the users are inundated by the details of tokens.

1.3 Abstraction as a Tool

In software engineering and computer science, abstraction [8] is a tool that allows programmers to think on a certain level of complexity, hiding the details not important to the problem at hand. We use abstractions to prevent overloading the end user with details when they care more about higher level concepts. Additionally, abstraction can be used to relate objects and help users focus on what the entity is vs. what it represents.

In the traditional financial industry, we have abstractions like the American stock market indices (e.g. S&P 500 [9], Dow Jones Industrial Average (DJIA) [10], etc.) that represent hundreds or thousands of stocks. In the insurance industry, we think of an insurance plan as a set of services we expect to be covered for when paying insurance premiums. We do not worry about the specific services when purchasing the plan. For cryptocurrencies, we can envision an abstract token or meta token, a single token representing a basket or portfolio of its underlying tokens.

2 {Set}

2.1 Overview

We introduce the {Set} (the braces evoke the mathematical set), an abstract token fully collateralized by its underlying tokens. While {Set} Protocol is designed to eventually support many blockchains and interoperate across blockchains, {Set}s are described in this document as smart contracts on the Ethereum blockchain as a superset of the ERC20 token standard.

2.2 Properties

- **ERC20 token + issue + redeem**: {Set}s are ERC20 [11] tokens with an issue and redeem function. ERC20 is a standard interface of functions and events that an Ethereum token contract must implement. ERC20 includes functions such as transfer, approve, totalSupply, and balanceOf. This means that {Set}s can be transferred, traded, approved to transfer, etc. - just like

any other ERC20 token. In addition, {Set}s have issue and redeem functions, which serve to convert between the {Set} token and its constituent tokens.

- **Fully collateralized**: {Set}s are fully collateralized by their underlying tokens (elements). This means that the {Set} contract trustlessly has custody of the elements and can only be manipulated through its exposed methods. {Set}s are created by taking the underlying tokens and issuing a new {Set} token that represents those tokens.

- **Redeemable**: {Set}s can be redeemed or traded for their elements. As {Set}s are backed by the underlying tokens, users can be certain that their {Set} can be traded for their components.

- **Specified tokens and units**: {Set}s are specified by a list of tokens and their respective quantities. Each token minted from the {Set} contract cannot deviate from the specified tokens or ratio of quantities, which is defined during the construction of the {Set} contract. As long as the desired tokens issued or redeemed matches the predefined {Set} weights, it is possible to issue and trade fractions of {Set}s.

- **Composable**: As long as the {Set}s conform to the ERC20 standard, {Set}s can be composed of other {Set}s. This makes it possible to have a single token represent a limitless number of other tokens without hitting the block gas limit and allows {Set}s to serve as a building block for other {Set}s.

- **Trustless**: All {Set} contracts are autonomous agents and function only as programmed. Unless deviating from the specification, {Set} contracts have no owners or administrators that can cause changes to the held collateral tokens.

2.3 Benefits

- **Save gas**: Without {Set}s, acquiring and transferring multiple ERC20 tokens requires paying transaction fees on transfers of each token. By transacting using {Set}s, users only need to pay transaction fees on a single transaction for the tokens they represent. For {Set}s that represent hundreds or thousands of tokens, the gas savings can be substantial.

- **Focus on higher-level concepts**: {Set}s allow users to think about higher-level concepts. Since {Set}s are composable, it is possible to build up multiple layers of abstraction. This makes {Set}s a very generalizable and a powerful

tool for building and reasoning about complex decentralized applications. This also serves as a tool for developers to relieve end users from the mental burden of intricate details that are not immediately important.

- **Underlying value**: Since {Set}s are collateralized by their underlying components, the intrinsic value of the {Set} can be determined by summing the value of its underlying components. We expect that {Set}s will be priced on token exchanges slightly above the cost of the underlying tokens, reflecting the cost of the issuer in effort and transaction fees to forge the {Set}.

- **No counterparty risk**: Other higher-level assets such as ETFs often trade at a discount, because these assets are not without counterparty risk. Because a trustless, autonomous smart contract holds custody to the underlying tokens, there is no third party that can fail to live up to its contractual agreements.

3 Use Cases

3.1 Index Token / ETF

In the traditional financial industry, market indexes are a collection of assets that track the overall movement in the equity markets. They represent the aggregate value of several securities and are created by combining the values of their underlying assets. Investors usually use indices to track the value of the market over time, gauge the markets financial health, and benchmark their own returns. Some of the most widely cited market indexes include the S&P 500, which tracks 500 largest companies having common stock on the largest US stock exchanges, and DJIA, which tracks 30 large publicly owned companies in the US. For investors, index fund investments are a passive form of fund management and are considered ideal core portfolio holdings for many peoples accounts. They usually have lower management expense ratios, allow for diversification, and lower trading costs for investors. In the cryptocurrency world, a {Set} is the natural tool for the representation of a collection of publicly listed tokens. As a market index that tracks tokens, {Set}s can be used for investment, tracking, market-making, and advanced derivatives:

- **Investment**: As an investment, {Set}s have many advantages including: low-cost, diversity, and low-maintenance.

  Without {Set}s, acquiring a basket of tokens can be prohibitively expensive in terms of fees and energy. Currently, investors who want exposure to a market index need to purchase the individual underlying tokens themselves. The

investor would need to 1) register and provide personal identification information for numerous exchanges, 2) pay maker-taker trading fees for each token purchase, and 3) pay network transaction fees to withdraw/transfer each token into their personal wallet. For example, if an investor wants to purchase tokens that represent prediction markets (e.g. composed of Augur and Gnosis), the investor would need to purchase each token individually from a multitude of exchanges that support these tokens. With {Set}s, the investor could purchase a single representative token from a single exchange, greatly reducing trading fees and saving time.

Owning {Set}s allows investors to diversify and manage their risk in any particular asset. {Set}s makes it easy for retail and institutional investors who do not know which individual crypto assets to purchase to get exposure to the entire market or subsector. For example, an investor who wants to get exposure to the top 10 tokens for long-term investment can invest by purchasing a single {Top10} set composed of the top 10 ERC20 tokens from his/her favorite exchange. For a desired {Set} that does not exist, the investor can invent / create their own {Set}.

Because {Set}s are passively managed and not meant to be actively traded, {Set}s are low-maintenance, only requiring infrequent rebalances. As the price of the underlying assets fluctuate, traditional market indices are rebalanced or updated by adjusting the collection of assets that are included in the index. This is a practice that is usually done quarterly, semi-annually, or annually. {Set} investors can easily rebalance their portfolios by trading for an updated {Set} that reflect the current collection of tracked tokens.

- **Tracking**: {Set}s are well suited to be a measurement of value for the token market. {Set}s can be used by investors and analysts to describe the market and compare returns on investments. Our current tools for tracking are quite limited. CoinMarketCap [1] only provides an aggregate market capitalization of cryptocurrencies. Current tools are also unstandardized. {Set}s can represent a construct that the industry relies on for evaluating the health of cryptocurrencies and performing macro analyses.

- **Market-Making**: Under the hood, {Set}s are most similar to exchange-traded-funds (ETFs). An ETF is a type of fund that owns the underlying assets and divides ownership into shares. Supply of ETF shares is regulated through creation (trading of underlying assets for ETF shares) and redemption (trading of ETF shares for the underlying assets). Creation and redemption

involve very specialized investors known as authorized participants (AP). The AP assembles the required portfolio of assets and turns that basket into the fund for newly created ETF shares. {Set} issuance and redeem mechanics are synonymous with the ETFs creation and redemption, and thus there will be the equivalent of APs in this system that will be issuing and redeeming {Set}s.

In the token world, there will be players who serve as liquidity-providers by taking advantage of mispricings in the market. Because the price of {Set}s fluctuate over time, there will be deviations of prices from the value of underlying assets. Traders can take advantage of these opportunities by issuing and redeeming {Set}s or by trading for other {Set}s.

- **Advanced Derivatives**: It is possible to create sophisticated financial products such as basket default swaps, collateralized debt obligations, or basket of futures by combining{Set}s with primitives such as derivatives (e.g. dy/dx [12]) and loans (e.g. dharma [13]). These instruments allow investors to manage risk and allow improved price discovery of tokens.

3.2 Processing Files

Let's say a developer wants programmatically to process a file in a MapReduce [14] fashion on computers all over the world using Golem and then store it in a decentralized manner using Storj. Using a decentralized exchange protocol such as 0x protocol, the developer can programmatically purchase a single {ComputeStore Token} which is composed of 100x Golem and 100x Storj tokens using wrapped Ether, ether that has been converted into the ERC20 specification. Then the developer can call the redeem function in the {ComputeStore Token} contract to get the underlying Golem and Storj tokens for usage on the Golem and Storj networks. Any unused tokens can be traded, sold, or issued into a new {Set}.

3.3 Health Care Package

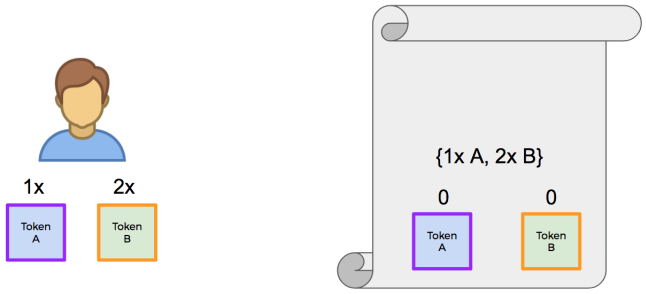
Imagine that a healthcare institution sells health care packages to individuals. These packages include 1x emergency room trip, 1x checkup, 2x dentist visit, and 1x eye doctor visit. The institution creates a token for each represented procedure. Then the institution creates a new {Set} called Health Token (HEALTH). Individuals who purchase a healthcare plan are issued HEALTH which could then be redeemed for its underlying tokens. With HEALTH, individuals can focus on purchasing a package of services instead of worrying about the details of individual services. When an

individual visits a dentist, they pay the dentist using the accepted dentist token. When an individual visits a doctor, they pay the doctor using the accepted checkup token. For any tokens/procedures that are not used, the individual could sell them to other users who need extra units of service.

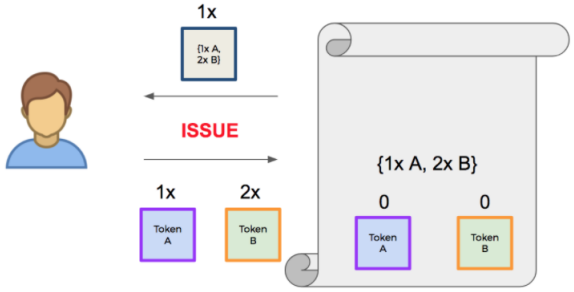
3.4 Other Use Cases

Please see Appendix 1 for additional use cases.

3.5 Issuance Demonstration

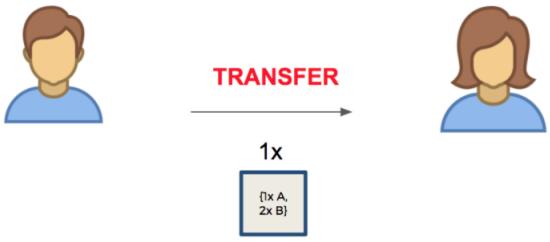


The newly instantiated {1x A, 2x B} contract is a {Set} that is composed of two tokens A and B, of quantities 1 and 2 respectively. The user holds 1x Token A and 2x Token B.

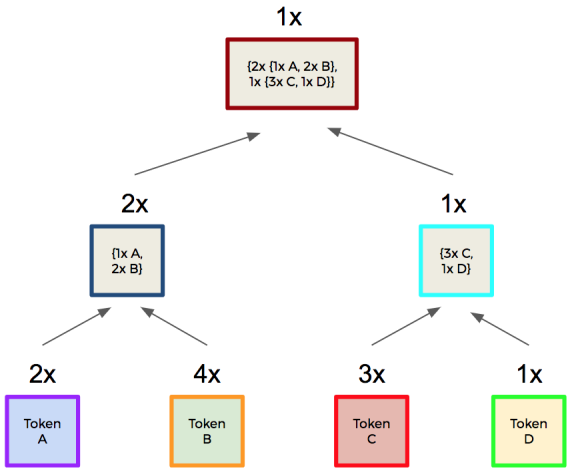


The user calls the issue function on the {1x A, 2x B} contract, sending to the contract 1x Token A and 2x Token B. The {1x A, 2x B} contract now holds 1x Token A and 2x Token B and gives the user a {1x A, 2x B} token in exchange for

the tokens received. This function is an atomic swap - reverting if any portion of the multi-step transaction has failed.



Since the token is an ERC20 token, the user can transfer, exchange, or sell the token to another user. It is also possible to issue and trade fractions of {Set}s.



It is possible to create {Set}s of {Set}s (reminiscent of power sets [14]) or {Set}s composed of other {Set}s. For example, the 2x {1x A, 2x B} can be combined with 1x {3x C, 1x D} to create {2x {1x A, 2x B}, 1x {3x C, 1x D}}. This composability makes it theoretically possible for a single {Set} to represent every single existing token.

4 Smart Contract

The {Set} contract is a specification implemented as an Ethereum smart contract [16], written in the Solidity programming language. The {Set} contract inherits from the StandardToken [17] contract and has three additional functions, which are the constructor, issue, and redeem.

4.1 Underwriting (constructor)

Underwriting is the process of creating a new {Set} contract. Anybody can create a {Set} by deploying a new {Set} contract that follows the {Set} specification to an Ethereum network. The constructor function is only called once during deployment. The required constructor parameters are below:

Table 1: Constructor Parameters

Parameter	**Type**	**Description**
tokens	address[]	A list of ERC20 token addresses
units	uint[]	A list of quantities for each token

There are no restrictions to how many different ERC20 tokens can be included, aside from non-duplicate address entries, the transaction gas limit, and data input limits. Since {Set}s are ERC20 tokens, address inputs of other {Set}s are valid. Deploying a {Set} creates a clean-slate contract with 0 tokens.

Example Usage:

Alice wishes to underwrite a basket of decentralized exchange tokens (DEX) including 0x (ZRX), Kyber Network (KNC), and Airswap (AST) tokens. Alice specifies the tokens she wants to create in an array of token addresses and the quantity of each token in an array. Next, Alice deploys a DEX smart contract to the network with the specified parameters ([0x...ZRX, 0x...KNC, 0x...AST], [1, 1, 1]). Who creates these contracts? Anybody who wants to present a higher level of abstraction to their end users. We envision that anybody who wants to list new {Set}s would create new {Set}s, given there is sufficient demand. Therefore, the abstraction creator would be the one bearing the gas cost for doing so.

4.2 Token Issuance (issue)

Token issuance is the process of generating new tokens from a {Set} contract. Given the {Set} contract has been deployed, anybody can call the contract's issue function to convert a specified mix of ERC20 tokens into a token that represents its underlying parts. It is possible to batch-issue, making it efficient (in term of gas) to issue a large number of new tokens. There only exists as many {Set} tokens as there are tokens issued less redemptions. The issue function parameters are:

Table 2: Issue Function Parameters

Parameter	Type	Description
quantity	unit	The quantity of {Set}s to issue

There are 7 steps to issuing a {Set} token:

Figure 1: {Set} Issuance Process

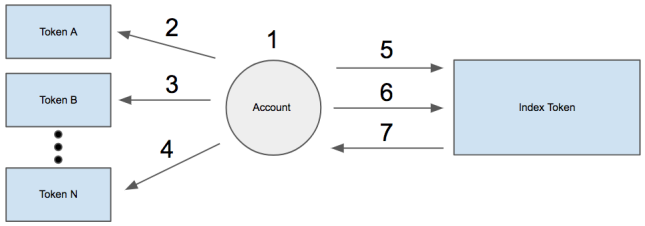


Figure 1 steps explained

1. Issuer decides the quantity of {Set} tokens to issue
2. Issuer calls Token A's approve function for the specified unit of Token A required multiplied by the quantity desired to issue
3. Issuer calls Token B's approve function for the specified unit of Token B required multiplied by the quantity desired to issue
4. Issuer continues to call approve for the correct amount to all remaining tokens
5. Issuer calls the {Set} Contract's issue function with the desired quantity of tokens to issue

6. The {Set} contract transfers the required quantities of tokens to the contracts. If any transfer is unsuccessful, the whole transaction is reverted

7. If the previous step was successful, the contract increments the corresponding quantity of tokens to the issuer. All tokens sent to the contract are held trustlessly as collateral for potential future redemption

Example Usage:

Alice now wishes to convert ZRX, KNC, and AST tokens she holds on her account into a Decentralized Exchange (DEX) token. Alice gathers the desired sums of ZRX, KNC, and AST and authorizes the contract access to her tokens. Then she calls the issue function on the DEX Smart Contract, and the contract automatically transfers the specified tokens to itself as collateral and sends to Alice a DEX token.

4.3 Token Redemption (redeem)

Token Redemption is the process of converting a {Set} into its underlying component tokens. Redeeming tokens reduces the token supply of {Set} tokens in the contract. The redeem function parameters are:

Table 3: Redeem Function Parameters

Parameter	Type	Description
quantity	unit	The quantity of {Set}s to redeem

There are 4 steps to redeeming a {Set} token:

Figure 2: {Set} Redemption Process

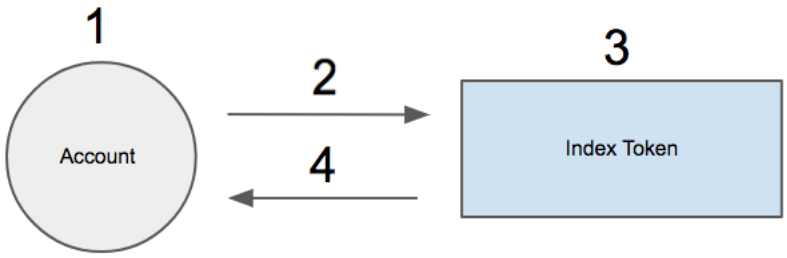


Figure 2 steps explained

1. Issuer decides the quantity of {Set} tokens to redeem
2. Issuer calls the redeem function with the quantity to redeem
3. The contract checks to ensure that the user has enough tokens. If not, the function reverts.
4. The contract transfers the underlying tokens to the sender and then decrements the sender's index token balance, effectively destroying the tokens and reducing the total supply of tokens.

Example Usage:
Alice now has a DEX index token and now wishes to convert her DEX index token back into ZRX, KNC, and AST tokens - because she wishes to use them to pay for fees or to vote on governance. Alice calls the redeem function with a desired quantity of tokens to redeem as a parameter. The redeem function destroys Alice's DEX tokens and transfers the corresponding number of ZRX, KNC, and AST tokens to Alice's account.

5 Future Infrastructure

5.1 Registeries

The {Set} Registry smart contract is a singleton contract that allows creation and tracking of {Set}s. The registry can create {Set} contract instances and register, which allows registration of an existing {Set} contract. In a registry, {Set}s are unique by the Keccak256 hash of the tokens included, units included, name, and symbol. {Set} registries can be specified to only include certain types or mixes of assets. These allow easy tracking and monitoring of existing contracts and exploration of {Set} types.

- {Set} registries could be created for a subset of {Set}s that contain tokens of equivalent types, but with different weights
- {Set} discovery and search would be done through lookups of the registries public, constant variables and functions, which are all free (non-gas charging) operations
- Registries can be created by any service-provider who needs to manage or keep track of a particular collection of {Set}s

- Multi-level registries are eventually necessary, and individual {Set}s are searchable using tree traversal algorithms. There are going to be registries for each basket of tokens, registries for related tokens, and registries for other registries

5.2 Javascript Library

Since many {Set} operations follow traditional set theory in math, there will be a need for conducting basic set arithmetic operations including superset, subset, complement, unions, etc. We also desire to build tools that make it easy for programmers to programatically trade {Set}s using decentralized exchange protocols such as 0x, Kyber Network, or Airswap.

- We envision that arbitrageurs and traders would take interest in tools that allow for efficient movements between {Set}s.

6 Existing Work

Existing work done around the idea of token abstraction has mostly been in the area of index tokens, products or services that represents a basket of tokens for investment purposes.

6.1 Individually Managed Index Tokens

Individually crafted token indexes such as HOLD10 (BitWiseInvestments) [18] are rigid and do not allow individuals to express their specific desired portfolios. These instruments are often centralized and curated by an individual team. For managed index funds, there is still counterparty risk - as another party is doing the work of aggregating and managing investments. {Set} aims to allow users to create any {Set} they desire without any counterparty risk. {Set}s can also be utilized by investment managers to reduce fees, group related investments.

6.2 Prism

Prism [19] is a index token project focused on using synthetic contracts collateralized with Ethereum, instead of the underlying base currencies. The benefit of this is that it is possible to represent non-ERC20 tokens on the ethereum blockchain and allows users to place bets on illiquid or even not yet issued tokens. Prism utilizes an Oracle system to update the price / underlying value, presenting a point of centralization.

In addition, Prism charges monthly fees (currently set at 0%) and closing fees and has an expiration date (like a derivative). Instead, {Set} aims to be unopinionated and as general as possible.

6.3 Bancor

Bancor's Smart Token [20] is very generalized and has many similarities with {Set}. Bancor's Smart Token allows for the swapping of underlying tokens for a new Smart Token. The main difference with {Set} is their autonomous conversion function, which allows swapping of any one of its composing tokens, which all Smart Tokens abide to. Thus, Smart Tokens are not atomically backed by its underlying tokens.

{Set} aims to a primitive on top of an ERC20 token, without any opinions about token price or liquidity. Also, {Set} aims to allow users and application-level logic on top of {Set} to take care of price discovery and liquidity. In addition, {Set}s are atomically backed by their underlying tokens, which guarantees a tokens underlying value.

6.4 0x Protocol

0x protocols token abstraction [21] is a process to obfuscate smart contract interactions between application tokens and the blockchain. The benefit is that application developers could create decentralized applications (dpps) that can utilize many tokens by only sending ether instead of having to aggregate many tokens individually. 0xs token abstraction and {Set} are highly complementary:

- **Single {Set} Purchase For Application Usage**: 0x requires developers to build functionality into dapps to pull/aggregate orders from relayers off-chain for the required tokens before batch filling those orders to acquire the tokens. Using {Set}s, users could directly acquire all the tokens needed for a decentralized application by purchasing a single {Set} and then redeeming the underlying tokens for usage. This is less work for developers and results in less trading / relayer fees.

- **Synchronous Token Acquisition and {Set} Issuance**: Using 0x, {Set} token issuers could very easily issue new {Set} tokens even if they don't already own the constituent tokens. A dapp could use 0x to aggregate orders for individual tokens, batch fill them to acquire the tokens, and then issue a {Set} token that represents all of the tokens. All this could be facilitated by a

Javascript library or a smart contract integration that makes it a single atomic transaction for batch filling the orders and issuing {Set}s.

7 Ongoing Research

Because {Set}s are such a new and unexplored asset class, there are many outstanding areas and questions of research. We expect that more wrinkles and complexities will become unveiled as we continue active research on this topic. Some current unanswered technical questions include:

- During construction, what checks are done by the {Set} smart contract and what do we defer to the developer?
- It is possible for contracts to be composed of tokens recursively. The problem is that the value of these "infinite" tokens would be impossible to determine, causing infinite loops in value aggregating programs. Careful development of cycle-detection algorithms is required to deal with this.
- What do we need to guarantee that lower-level tokens cannot be manipulated by their owners or changed by monetary policy? Developers need to be careful about choosing tokens that are trusted. Changes to underlying tokens could potentially affect the ability of higher-level tokens to function properly. What should the policy be for tokens that have forked or have upgraded?
- Issuing multiple tokens allows a central party to create a bunch of tokens for distribution for other parties to use. What is the right UI/UX to make this easy for users?
- How should registries and multi-level registries be designed?
- How can we design smart contracts that consume the underlying tokens of {Set}s and provide refunds for tokens not used?

7.1 Summary

We introduce the idea of a {Set}, which is a single token that represents a basket of tokens. {Set}s are a powerful tool for allowing us to hide away the details of underlying tokens and focus on higher-level concepts. It represents a new asset class that allows more efficient transactions, is fully collateralized, and is trustless. {Set}s

serve as a fundamental building block for composing more complex financial instruments. {Set}s are a primitive and serve as a powerful tool that allows developers and product-creators to build increasingly complex decentralized applications that are user-friendly and intuitive. There is a limitless number of {Set} use cases.

7.2 Acknowledgements

We want to express our gratitude for all those who have helped out in the formation of this paper. Thank you to Inje Yeo, Lane Rettig, Jeff Chang, Matthew Liu, Yondon Fu, Linda Xie, Antonio Juliano, Nik Kunkel, Michael Karnjanaprakorn, Sukhveer Sanghera, Michael Wong, and others who have helped me clarify my thoughts and perform copy-editing. Thank you for the organizers of EthWaterloo for hosting a hackathon in October 2017 where the idea of the {Set} was conceived. Thank you to all those at Consensys and my peers at Consensys academy that helped refine the ideas around {Set}.

8 References

1. Coinmarketcap. https://coinmarketcap.com/ (Accessed November 2017)

2. Russia is considering an official cryptocurrency called the 'CryptoRuble'. http://www.businessinsider.com/russia-is-considering-an-official-cryptocurrency-2017-10

3. emCash is Dubais First Official State Cryptocurrency. https://www.cryptocoinsnews.com/emcash-dubais-first-official-state-cryptocurrency/

4. PBoC Digital Currency Director Calls for Centralized State Cryptocurrency. https://www.coindesk.com/pboc-digital-currency-director-calls-centralized-state-cryptocurrency/

5. Enterprise Ethereum Alliance. https://entethalliance.org/hewlett-packard-enterprise-47-organizations-join-200-member-strong-enterprise-ethereum-alliance/

6. Goldman Sachs exploring bitcoin trading operation. https://www.cnbc.com/2017/10/02/goldman-sachs-exploring-bitcoin-trading-operation-report-says.html

7. Ethereum Blockchain in Jordan Is Changing How the UN Delivers. Aid https://borgenproject.org/ethereum-blockchain-in-jordan/

8. Abstraction. https://en.wikipedia.org/wiki/Abstraction_(software_engineering)

9. S&P500. https://us.spindices.com/indices/equity/sp-500

10. DJIA. https://en.wikipedia.org/wiki/Dow_Jones_Industrial_Average

11. ERC20 https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md

12. dy/dx. https://dydx.exchange/

13. dharma. https://dharma.io/whitepaper/

14. Power Set. Wikipedia. https://en.wikipedia.org/wiki/Power_set

15. MapReduce. https://en.wikipedia.org/wiki/MapReduce

16. Ethereum Smart Contract. http://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html

17. Standard Token Contract. https://github.com/OpenZeppelin/zeppelin-solidity/blob/master/contracts/token/StandardToken.sol

18. Bitwise Investments. https://www.bitwiseinvestments.com/

19. Prism. https://info.shapeshift.io/blog/2017/05/21/introducing-prism-worlds-first-trustless-portfolio-market-platform

20. Bancor. https://www.bancor.network

21. 0x project. https://0xproject.com/pdfs/0x_white_paper.pdf

9 Appendix

1.
   - Carbon Neutral Token: A {Set} that is composed of a barrel of oil and carbon tax credits.
   - Bundling for Gifts: A {Set} composed of a bundle of tokens to gift to their friends and family.
   - Healthcare Identity: A {Set} that aggregates one's healthcare information (represented as individual tokens) for transfer to healthcare providers when requested.
   - Diversified Portfolio of Bonds: A {Set} that aggregates bonds of various interest rates and expiration dates for investment.
   - Student Loan Token: A {Set} composed of tokens that can restrictively be redeemed for tuition and housing.
   - All-Inclusive Vacation Package: A {Set} representing a tourist package which aggregates flights, hotel stay, meal vouchers, tours, and local transportation.
   - Loyalty programs: A {Set} that represents a users loyalty rewards programs including airlines, hotels, car rental programs, etc.