Bskt: a smart contract for creating decentralized token portfolios v0.3.4

D.Que, N. Dalal, Q. Pham, J. Tong

***Abstract*— Bskt is a generic smart contract that creates decentralized token portfolios. Bskt facilitates the bundling and unbundling of a collection of Ethereum tokens in the form of an ERC20 token. Owners of the token have a direct claim on the underlying tokens.**

These new tokens can be created by anyone who surrenders the underlying tokens and redeemed by anyone who owns issued tokens. Bskt allows investors to diversify their exposure to tokens in the Ethereum ecosystem cheaply, without adding custody risk.

I. Introduction

The Bskt contract provides a container that holds a collection of underlying Ethereum ERC20 tokens [1] with no custodial risk. Bskt can also be used as a general framework for containers that hold any cryptographic tokens.

In early 2018, there were more than 600 Ethereum-based ERC20 tokens with a combined market cap of over $60 billion [2]. Demand for cryptographic tokens will likely continue to increase with the innovations in creating cryptographically-native "security" tokens, and the rush to tokenize real world assets, from real estate to stocks.

The proliferation of cryptographic tokens has made it important to develop a structure that easily holds these tokens.

Bskt has many similarities with Exchange Traded Funds (ETFs) [3], Unit Investment Trusts (UITs) [4], and Real Estate Investment Trusts (REITs) that hold trillions in assets [5] worldwide. As in the real world, these structures make it easy to trade diverse assets and reduce transactions costs.

ETFs specifically were designed to allow non-institutional investors to obtain and transfer a more diversified portfolio of equities cheaply. This Ethereum software solution provides similar benefits.

II. Overview

Bskt is an Ethereum contract which holds custody of underlying assets on behalf of token holders. At its core, the ERC20-compliant Ethereum contract allows one to:

- **instantiate** a custom Bskt with both pre-specified underlying tokens and a selected proportion,
- **create** Bskt tokens in exchange for surrendering underlying tokens,
- **redeem** Bskt tokens for the underlying tokens, and
- **transfer** the ownership of Bskt tokens.

Bskt is not a new blockchain or protocol, but an Ethereum contract that can be configured to hold any number of underlying ERC20 tokens (see an example in Fig 1). The newly created tokens fluctuate in value based on the value of the underlying tokens.

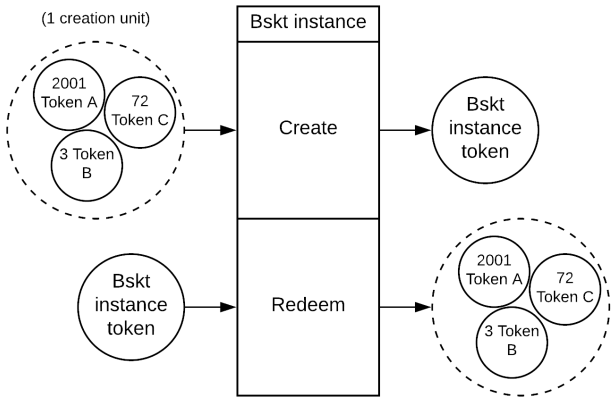

Fig. 1: Illustration of Bskt usage. This shows an example of a single deployed Bskt composed of Token A, Token B, and Token C. This contract allows the creation and redemption of Bskt tokens.

Bskt also formalizes an open atomic swap standard to allow upgrades and portfolio adjustments. This requires users to opt-in, and enables any developer to offer bug fixes or changes to the underlying composition.

Since Bskts are ERC20 tokens, new Bskts can be composed of various underlying Bskts.

A. Advantages

Through the use of Bskt, holders get the following benefits:

- **Portfolio diversification**: A number of different tokens can be combined to create a diverse portfolio.
- **Accessibility**: Tokens are often spread across multiple exchanges, and not available to all investors. Bskt makes it easy to access a wide variety of tokens.
- **Low transaction fees**: Purchasing and then transferring all the tokens off exchanges is costly; though there is a cost to creating a Bskt instance, creating a tradeable unit reduces future costs.
- **Ease of transfer**: Moving many different tokens is a poor user experience and increases the risk of mistakes.
- **Fractional transfers**: Bskt can be used to divide an expensive, non-tradeable token into manageable units.

Bskt has some similarities to traditional asset structures like ETFs, but has some distinct advantages:

- **Decentralization**: Since custody is managed by a smart contract, a third-party can't control or steal the underlying tokens.
- **Transparency and trustlessness**: Bskt's holdings are fully transparent and publicly readable on the

blockchain. It is easy to identify an under-collateralized Bskt.

- **Ease of portfolio creation**: Bskts providing exposure to different sets of underlying tokens are easy to create without much overhead.
- **Cryptographically native**: Bskts are a blockchain asset that blockchain-based smart contracts can build on top of directly, unlike traditional financial instruments.
- **Democratized creation**: Creating and redeeming in traditional asset structures (like ETFs) is only accessible by a handful of wealthy, specialized investors. One Bskt creation unit can be orders of magnitude smaller than the size needed for traditional asset classes.

III. How it works

A. Terminology

ERC20: ERC20 is a token standard. Any smart contract which implements the standard's interface is considered an ERC20. The key methods relevant to this paper are described:

- `transfer` allows an owner to send tokens to an address.
- `approve` enables an owner to allow another spender address to transfer their tokens.
- Approved tokens are transferred with `transferFrom`.

Base unit (BU): The atomic, least divisible unit of a token or coin. The base unit for Ethereum is wei.

Decimals: Refers to the ERC20 `Decimals` property which determines the base unit of a token. `Decimals` is typically set to 18 but can vary. For Bskt `Decimals` is always 18, making the base unit 10^{-18}.

Bskt contract vs Bskt: Bskt contract refers to the generic contract which can be deployed with parameters to create Bskt instances. Bskt instances can represent different portfolios.

Creation unit: The least divisible amount required for creation and redemption. Bskts must be created and redeemed in multiples of the creation unit.

B. Summary

The Bskt contract can be easily configured for a specific list and proportion of underlying tokens. A deployed instance of the contract is a *Bskt instance*. Bskt follows the standard ERC20 token standard.

ERC20 token name	Token address	Quantity per creation unit (BU)
Token A	0x86fa0498...78ecfdb0	2001
Token B	0xf230b790...3bed42e2	72
Token C	0xd26114cd...dB8A0C07	3

TABLE I: An example Bskt instance. This table notes key values to specify a new Bskt.

C. Key actions

Instantiation: To create a new Bskt instance, a user specifies the addresses and proportions of underlying Ethereum tokens. Once deployed, anyone can create or redeem using that instance of the Bskt.

Creation: The `create` function allows users to mint tokens in exchange for surrendering the specified quantities of underlying tokens. This function requires a multiple of a pre-specified "creation unit" to be created. Tokens from one Bskt instance *cannot* be used with another Bskt instance, as they may have different underlying tokens.

The create process works as follows:

1) Determine how many Bskt tokens to create based on your desired amount. This is generally automated on a website that can read a Bskt instance's initialization parameters, mapping a requested Ether amount to how many underlying tokens will be needed.
2) Acquire the underlying tokens (usually bought on exchanges) and transfer to an address you control.
3) Call the ERC20 `approve` function for each underlying token, allowing the Bskt to access the appropriate amount of each token.
4) Call the Bskt's `create` function.
    - Uses the ERC20 `transferFrom` function, then mints Bskt tokens for the creator.

Our expectation is that only advanced users and arbitragers will create or redeem bkst instance tokens, with most users simply transfering them. Open source tools will be available to make the create/redeem process simple.

Redemption: The `redeem` function first burns tokens according to the amount specified, then transfers the underlying tokens to the redeemer with ERC20's `transfer` function.

Redeem can optionally take an additional argument, `tokensToSkip` which specifies which underlying tokens to skip. This argument is valuable given the pausable nature of many Ethereum tokens today and for reducing the risk of ERC20 tokens maliciously interfering with Bskts.

For example, many popular tokens fail on transfer if paused. Their teams plan to permanently pause their tokens very soon when moving to their own blockchain. The following table shows some prominent tokens and their planned pause dates:

Name	`approve`	`transfer`	`transferFrom`	Pause date
EOS	pausable	pausable	pausable	Jun/01/2018
ICON	pausable	pausable	pausable	Mar/x/2018
Augur	ok	pausable	pausable	N/A

TABLE II: This table shows whether key ERC20 operations are pausable and the expected pause date for some prominent tokens.

Without the skip list, the failure of a single token transfer could permanently prevent the redemption of all other tokens. We mitigate this problem by allowing users to specify which tokens to skip. This prevents other tokens from being locked at the cost of locking skipped tokens. *The skip list feature should only be used when an underlying token is failing and preventing redemption.*

Pausing (owner only): The `create` action can be paused by a contract owner. This is intended to enable the deprecation of old Bskts, while still allowing holders to access the underlying tokens with `redeem`.

Extraction (future version, owner only): One or more underlying tokens may have to be removed from the initially created bundle. For example, underlying ERC20 creators may lock their token when they're ready to launch their native blockchain.

Rather than require all holders to redeem their tokens followed by the issuance of a new Bskt instance, the Bskt contract owner can decouple individual tokens from the bundle by extracting them (see Fig 2). This process ensures that Bskt holders maintain full custody at all times.

The outcome of this action is that holders hold two tokens: an updated token that excludes the extracted token, and the extracted token. The extracted token can be claimed and transferred elsewhere.

Excess funds collection (owner only): There are several scenarios where third-parties may leave excess funds in the contract:

- A user sends ERC20 tokens to the contract directly, rather than using ERC20's `approve` then Bskt's create.
- A user sends Bskt tokens to the Bskt contract directly, rather than using `redeem`.
- A user uses `tokensToSkip` in the redeem function, but these skipped tokens are able to be transferred later.
- A user sends Ether to the contract.

In all cases, the excess funds and tokens are available to the contract owner through the `withdrawExcessToken` and `withdrawEther`.

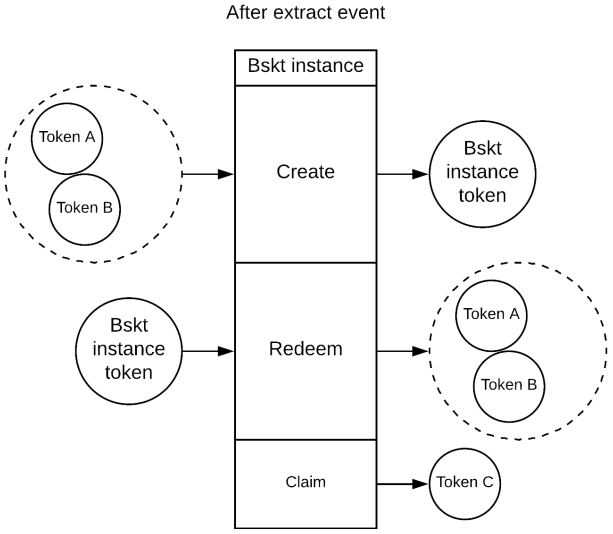

Fig. 2: After Token C has been extracted from a Bskt instance, it's no longer required to `create` and it isn't redeemable by `redeem`. A snapshot of ownership is recorded, so only those who owned Bskt tokens before extraction are able to claim Token C.

Atomic swap: Circulating Bskts may need to be changed to apply an upgrade, a security fix to the contract, or an adjustment of the token composition. To enable this, new Bskts can be exchanged for old Bskts (see Fig 3).

Contract owner actions: To ensure custody is held by token owners at all times, only specific actions have been reserved for the contract owner. In the Bskt contract, `create` can be paused to allow for deprecation, and excess funds can be withdrawn. The contract owner cannot collect underlying "reserved" tokens or prevent transfers.

Fig. 3: Bskt instance tokens are upgraded with an atomic swap facilitated by an upgrade contract. Old Bskt v1 tokens are exchanged for new Bskt v2 tokens.

D. Important parameters

Addresses: Bskts are configured by specifying the addresses of the underlying tokens.

Quantities: The numbers of underlying tokens required for one creation unit, in base units. This allows the setting of the initial portfolio allocation and determines the price for a single Bskt.

All quantities must be representable by an unsigned integer. Quantities are computed based on the creation unit, and are optimized such that Bskts retain as much divisibility as possible.

Creation unit: Bskts must be created and redeemed in multiples of a creation unit. The creation unit constraint is assigned to prevent fractional amounts of underlying tokens. This allows them to be defined as unsigned integers, and makes loss of precision impossible.

Bskts can be transferred in amounts as small as 1 base unit (10^{-18}), meaning fractions of a creation unit can be transferred. Unlike real world ETFs, the creation unit is generally a small number (often much less than $0.01).

The creation unit is computed based on these constraints:

- Target price
- Portfolio allocation
- Underlying token divisibility
- Underlying token price
- Requirement that all token quantities must be natural numbers
- Maximization of Bskt divisibility

The formula used to calculate the creation unit is:

$$best\ creation\ unit = 10^{18-\lfloor log(min_i(quantity_i)) \rfloor}$$

This is best illustrated with an example, shown in Table III.

IV. Implementation

Please note: As part of this whitepaper, we are releasing the Bskt contract v1 under the MIT License. All code and documentation is provided as is – and has the risk of bugs.

Token	Market cap	Price	Decimal	Allocation	Value per Bskt	Quantity per Bskt	Quantity per Bskt (BU)	Best creation unit	Quantity per creation unit (BU)
Token A	$5,027,187	$4.61	18	48.11%	$2.41	0.5217	5.22E+17	N/A	5,217,270,035,202
Token B	$2,739,853	$8.18	6	26.22%	$1.31	0.1602	1.60E+05	N/A	2
Token C	$2,681,325	$4.94	12	25.66%	$1.28	0.2598	2.60E+11	N/A	2,597,843
Totals	$10,448,365			100%	$5.00			1.00E+13	

TABLE III: This table shows an example derivation of creation unit with a target price of 5 USD. Market cap, price, and decimal are inputs based on the token. Allocation is the percent representation of the token in the portfolio, weighted by market cap. Value per Bskt is the value of the token required for 1 Bskt. Quantity per Bskt is the amount of the token required to achieve the value required. Quantity per Bskt (BU) is the same as the last column, but in base units. Best creation unit is computed from the previous column, and is optimized for Bskt divisibility. Quantity per creation unit (BU) is the final quantity required to constitute one creation unit. These values are the ones used to initialize a Bskt instance. See the example spreadsheet [6] to see the full formulas used.

Our implementation of Bskt is available on GitHub [7]. A test Bskt portfolio instance has been deployed to mainnet, and can be found in our repo.

V. Security

Since Bskt depends heavily on the ERC20 functionality of `approve`, `transfer`, and `transferFrom`, it's subject to how they're implemented. Often, teams that deploy ERC20 tokens make modifications to these functions.

When creating a new Bskt instance, deploying teams should be sure to audit all underlying token contracts to ensure no malicious or unexpected behavior is possible.

A. Denial of service

Due to the synchronous nature of `create` and `redeem`, if an operation fails for a single token, it prevents the entire operation from working.

This could be benign such as a token pausing in order to do a native token airdrop upgrade, or malicious such as an underlying token seeking to hold tokens hostage by locking them. After auditing the most popular token contracts, we've seen that the most plausible way for `transfer` or `transferFrom` to fail is if the token owners decide to pause it.

As a failsafe, `redeem` is protected with the skip list feature described earlier.

Unfortunately `create` can't be protected by a skip list. Thus if an underlying token becomes paused, it will no longer be possible to create new Bskts.

Both Bskt instance creators and holders should verify that the underlying tokens work with Bskt. Not all ERC20 tokens will work with Bskt.

VI. Future work

Asset tokenization: The tokenization of other blockchains and assets like securities and property will enable Bskt to create a greater variety of Bskt portfolios. For example, a Bskt could include Bitcoin, NEO, and Nano as well as ERC20 tokens.

Rebalancing: Bskt can codify rebalancing rules so the portfolio is adjusted on a periodic basis. Both active and passive management are possible. Users must opt-in once, but from then on don't have to perform any actions to have their Bskts rebalanced.

VII. Existing work

The idea of grouping assets is a common theme in finance and early cryptocurrency projects. In finance, ETFs and UITs are widely used products that make it easy for investors to purchase grouped assets cheaply and quickly. Our key contribution is providing a complete, tested Ethereum contract that bundles tokens, while recognizing some of the key failure states on Ethereum today.

A. Crypto20

Like Bskt, Crypto20 is an ERC20 token that represents ownership of underlying crypto tokens, including many non-Ethereum tokens. Unlike Bskt, custody of the underlying assets is held by a single party, requiring that the holder trust Crypto20.

B. {Set}

The {Set} Protocol provides a framework for bundling ERC20 tokens. It shares many similarities with Bskt, including a smart contract with creation and redemption methods, and custody held by the contract. Bskt's key additions are overcoming real-world usage challenges such as: 1) skip functionality when redeeming to mitigate pausing issues with underlying tokens, 2) extract functionality to remove tokens from an existing Bskt, 3) a framework for atomic swaps, and 4) method for deriving optimal creation units which allows for fractional amounts to be transferred.

References

[1] Vitalik Buterin Fabian Vogelsteller. *EIP 20*. URL: `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md`. (accessed: 03.13.2018).

[2] CoinMarketCap. *Token Market Capitalizations*. URL: `https://coinmarketcap.com/tokens/`. (accessed: 01.15.2018).

[3] Community. *Exchange traded fund*. URL: `https://en.wikipedia.org/wiki/Exchange-traded_fund`. (accessed: 03.13.2018).

[4] Community. *Unit investment trust*. URL: `%https://en.wikipedia.org/wiki/Unit_investment_trust`. (accessed: 03.13.2018).

[5] Ryan Vlastelica. *ETFs shattered their growth records in 2017*. URL: https://marketwatch.com/story/etfs-shattered-their-growth-records-in-2017-2017-12-11. (accessed: 03.13.2018).

[6] Cryptofin. *Example creation unit derivation*. URL: https://example.erc20twenty.com. (accessed: 03.21.2018).

[7] Cryptofin. *Bskt*. URL: https://github.com/cryptofinlabs/bskt. (accessed: 03.21.2018).