



(19) **United States**

(12) **Patent Application Publication**
STRADLING et al.

(10) **Pub. No.: US 2020/0258159 A1**

(43) **Pub. Date: Aug. 13, 2020**

(54) **SYSTEM AND METHOD OF PROVIDING A CONTRACT-CREATOR APPLICATION**

G06Q 50/18 (2006.01)

G06F 12/14 (2006.01)

G06Q 40/04 (2006.01)

G06Q 30/06 (2006.01)

G06Q 20/38 (2006.01)

(71) Applicant: **ShapeShift AG**, Zug, OT (CH)

(72) Inventors: **Adam STRADLING**, Berlin (DE);
Erik VOORHEES, Denver, CO (US)

(52) **U.S. Cl.**

CPC *G06Q 40/06* (2013.01); *G06Q 30/018*

(2013.01); *H04L 9/3236* (2013.01); *H04L*

67/26 (2013.01); *H04L 63/105* (2013.01);

G06Q 50/188 (2013.01); *H04L 9/3297*

(2013.01); *H04L 9/3268* (2013.01); *G06F*

12/1408 (2013.01); *G06Q 40/04* (2013.01);

G06Q 30/0641 (2013.01); *G06Q 20/3829*

(2013.01); *G06Q 2220/00* (2013.01); *H04L*

2209/56 (2013.01); *H04L 2209/38* (2013.01);

G06Q 40/00 (2013.01)

(21) Appl. No.: **16/864,684**

(22) Filed: **May 1, 2020**

Related U.S. Application Data

(63) Continuation of application No. 15/715,746, filed on Sep. 26, 2017.

(60) Provisional application No. 62/399,763, filed on Sep. 26, 2016, provisional application No. 62/453,416, filed on Feb. 1, 2017, provisional application No. 62/453,350, filed on Feb. 1, 2017, provisional application No. 62/453,379, filed on Feb. 1, 2017, provisional application No. 62/453,384, filed on Feb. 1, 2017.

(57)

ABSTRACT

Disclosed is a system and method of creating a smart contract on a blockchain. The approach includes receiving, from a user, one or more parameters via a user interface, each of the one or more parameters being associated with a creation of a customized smart contract. The approach also includes authenticating the one or more parameters via a public/private key associated with the user, and deploying the customized smart contract onto a blockchain in a provably honest fashion. The customized smart contract runs without a custodial risk and can be established between a first party and a second party with no third party holding custody of any assets associated with the customized smart contract.

Publication Classification

(51) **Int. Cl.**

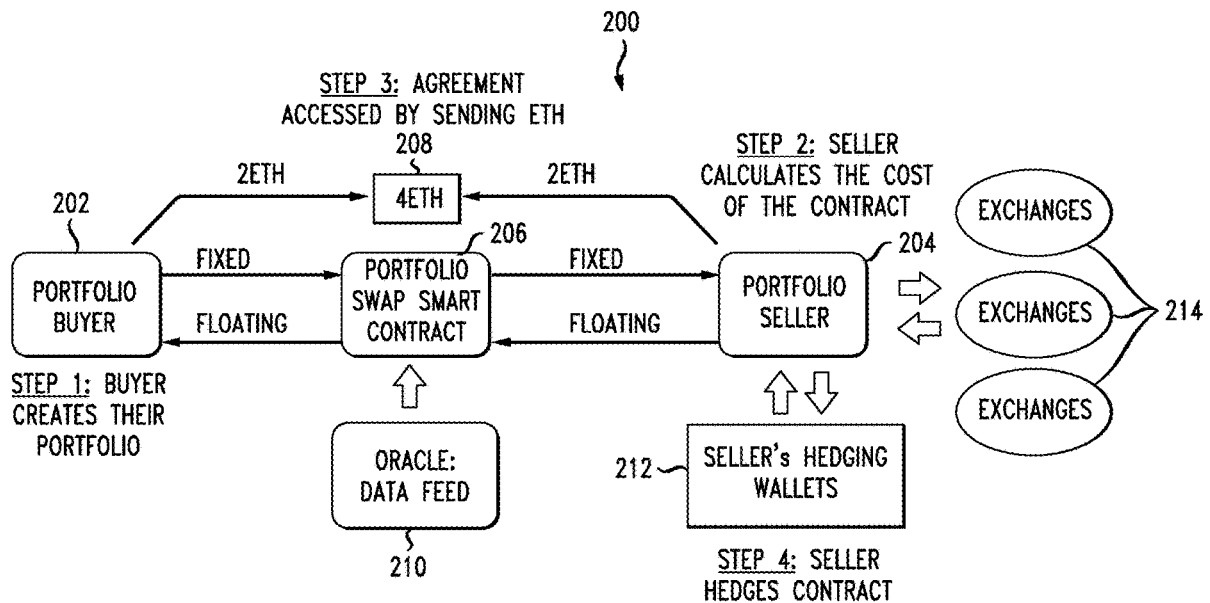
G06Q 40/06 (2006.01)

G06Q 40/00 (2006.01)

H04L 9/32 (2006.01)

H04L 29/08 (2006.01)

H04L 29/06 (2006.01)



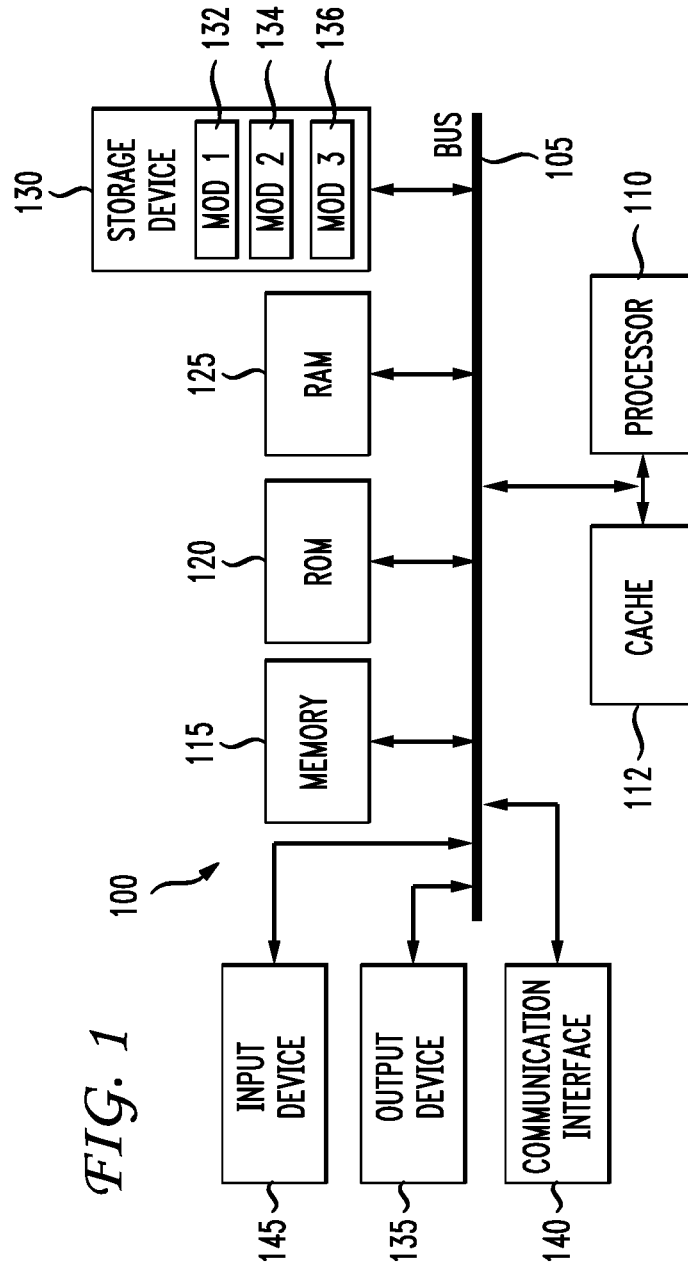


FIG. 1

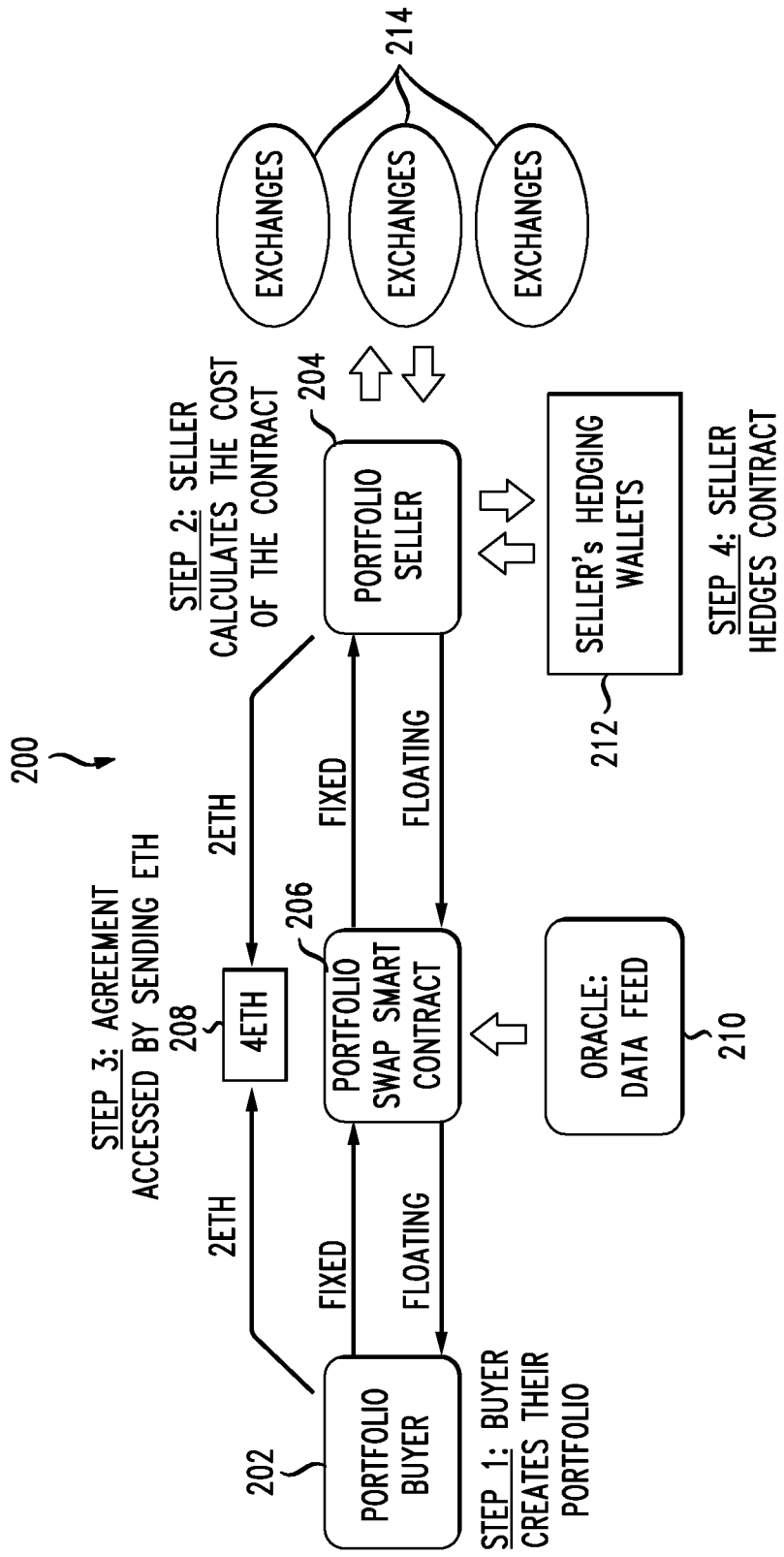


FIG. 2

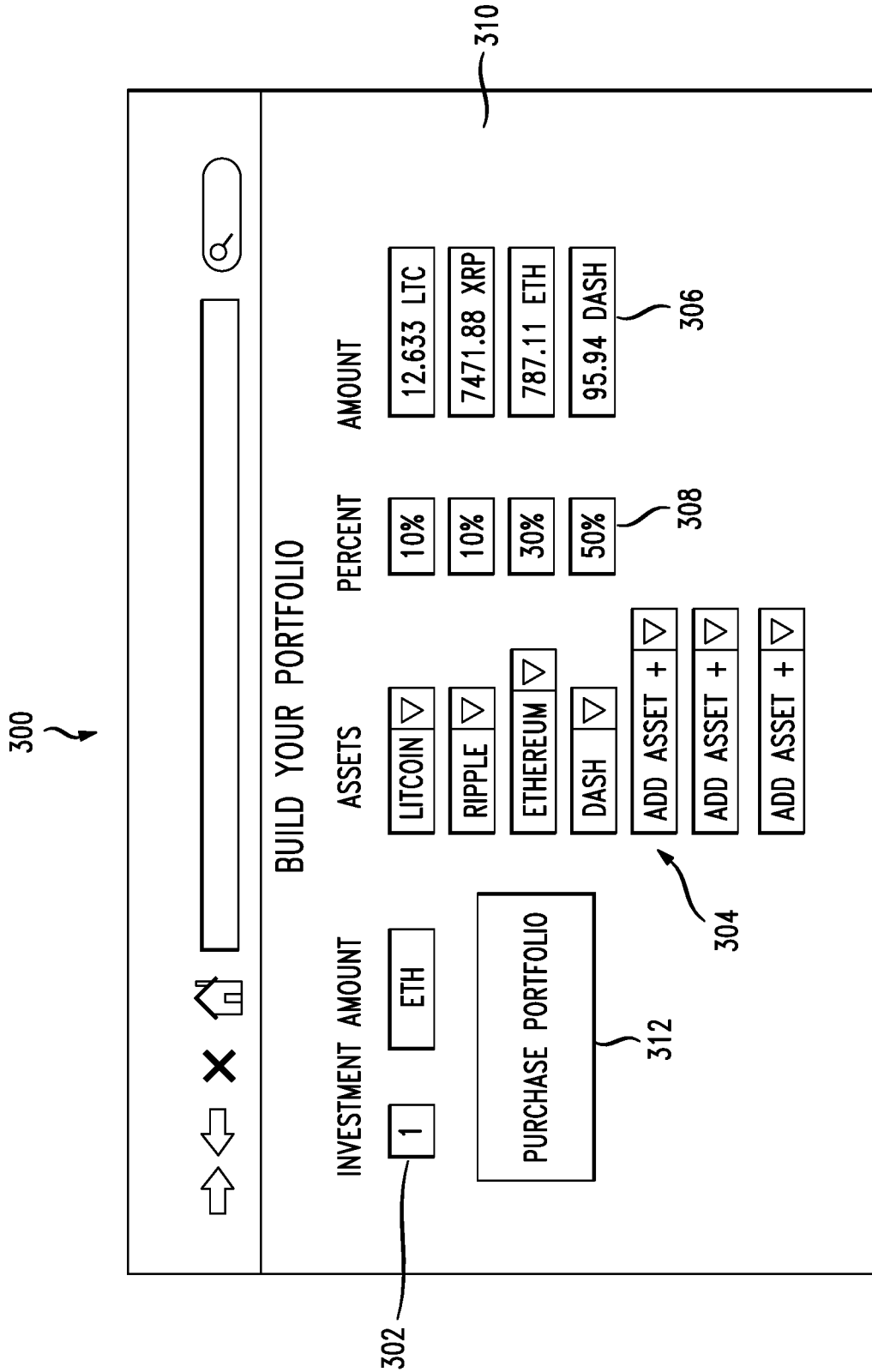


FIG. 3

400 ↙

↶
✕
🏠

🔍

402

PORTFOLIO PURCHASE SUMMARY

| PORTFOLIO VALUE | ASSETS | PERCENT | AMOUNT |
|-----------------|----------|---------|-------------|
| 1.0 BITCOIN | LITCOIN | 10% | 12.633 LTC |
| \$408.00 USD | RIPPLE | 10% | 7471.88 XRP |
| 1.0 ETH | ETHEREUM | 30% | 787.11 ETH |
| 1.0 DASH | DASH | 50% | 383.76 DASH |

BUY YOUR PORTFOLIO

PAYMENT DETAILS

PLEASE SEND 1 ETH TO:

30JmV3qfvL9SuYo34ihAf3sRCW3qSinyC

WARNING! WARNING! WARNING!

YOU MUST RETAIN THE PRIVATE KEYS OF THE ADDRESS FROM WHICH YOU MAKE THE PAYMENT. WHEN YOU CHOOSE TO SELL YOUR PORTFOLIO, THE BITCOINS WILL BE SENT TO THIS ADDRESS TO LEARN MORE READ OUR TERMS HERE.

404

YOU HAVE 12:15 MINS LEFT TO PAY:

FIG. 4

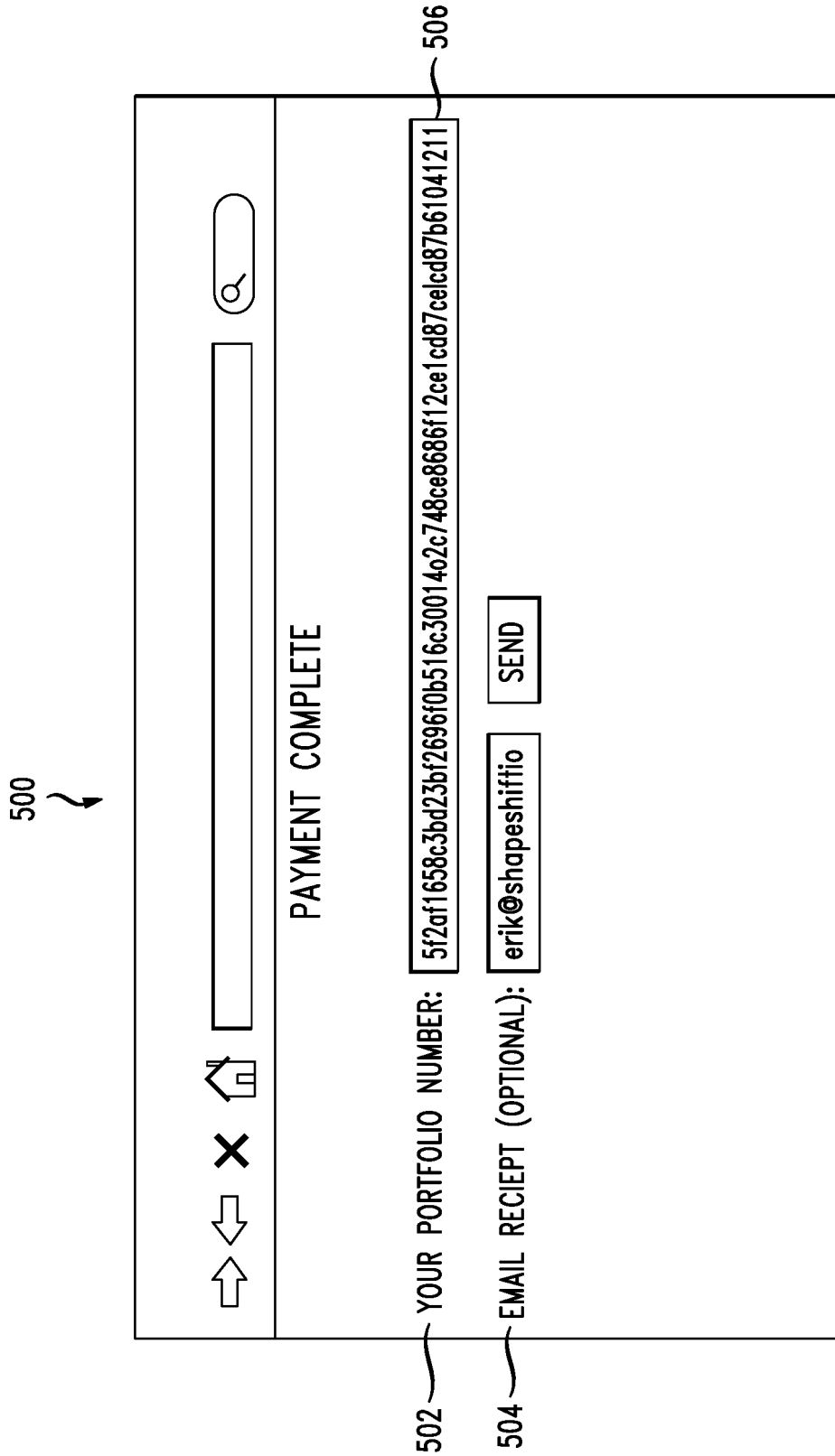


FIG. 5

600



↔
✕
🏠

🔍

PORTFOLIO SUMMARY

| CURRENT VALUE | INITIAL VALUE | ASSETS | PERCENT | AMOUNT | CURRENT RATE (/BTC) | GAIN/LOSS % |
|---------------|---------------|----------|---------|-------------|---------------------|-------------|
| 1.50 BITCOIN | 1.0 BITCOIN | LITCOIN | 10% | 12.633 LTC | 113.40 LTC | -10% |
| \$672.00 USD | \$408.00 USD | RIPPLE | 10% | 7471.88 XRP | 8222.36 XRP | +10% |
| 1 ETH | 1 ETH | ETHEREUM | 30% | 787.11 ETH | 262.37 ETH | +0% |
| 2 DASH | 1 DASH | DASH | 50% | 383.76 DASH | 95.94 DASH | +100% |

PORTFOLIO ACTIONS:

| | |
|---------------------|---|
| REBALANCE | ▽ |
| SHARE | |
| ADD TO LeaderBoards | |
| SELL | |

602

604

FIG. 6

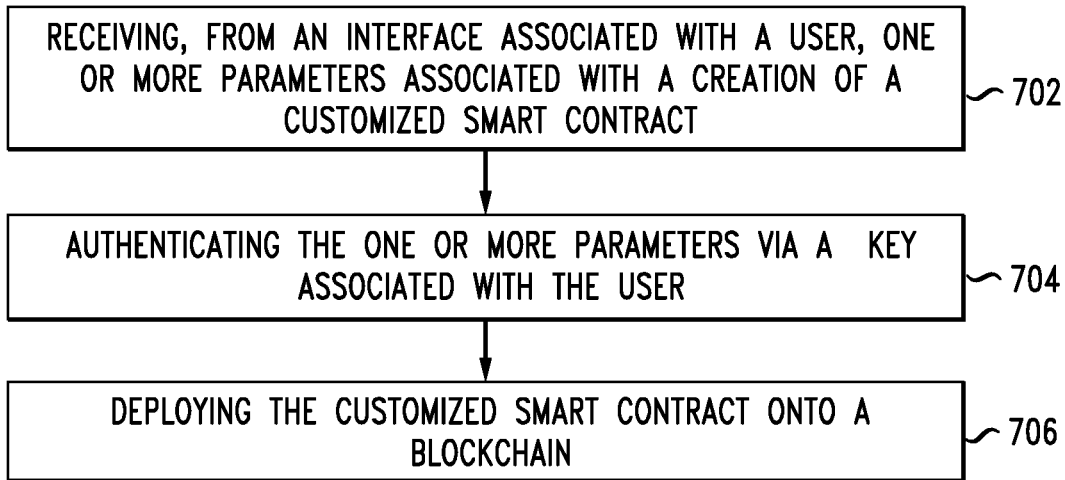


FIG. 7

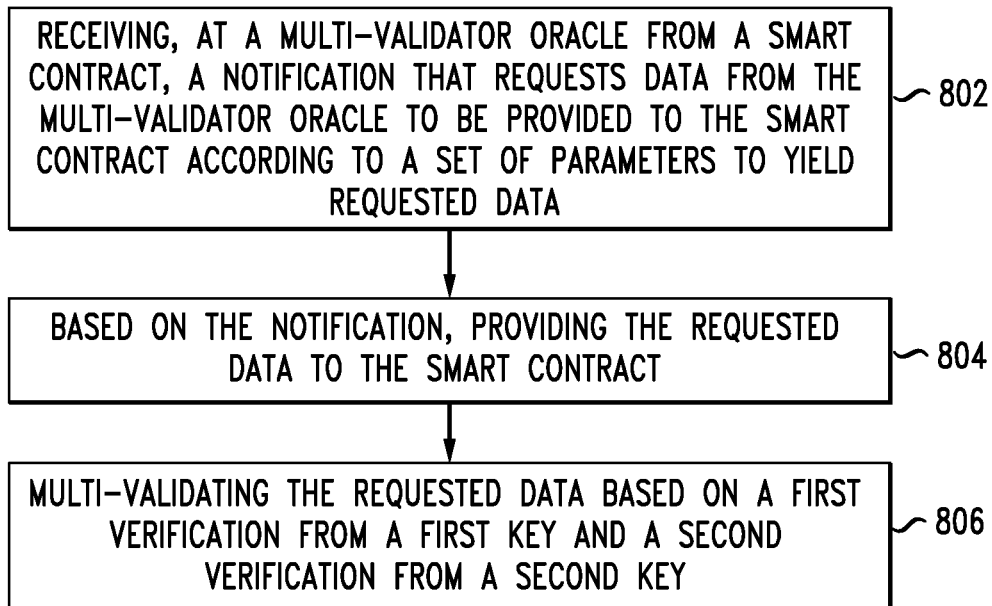


FIG. 8

900 ↙

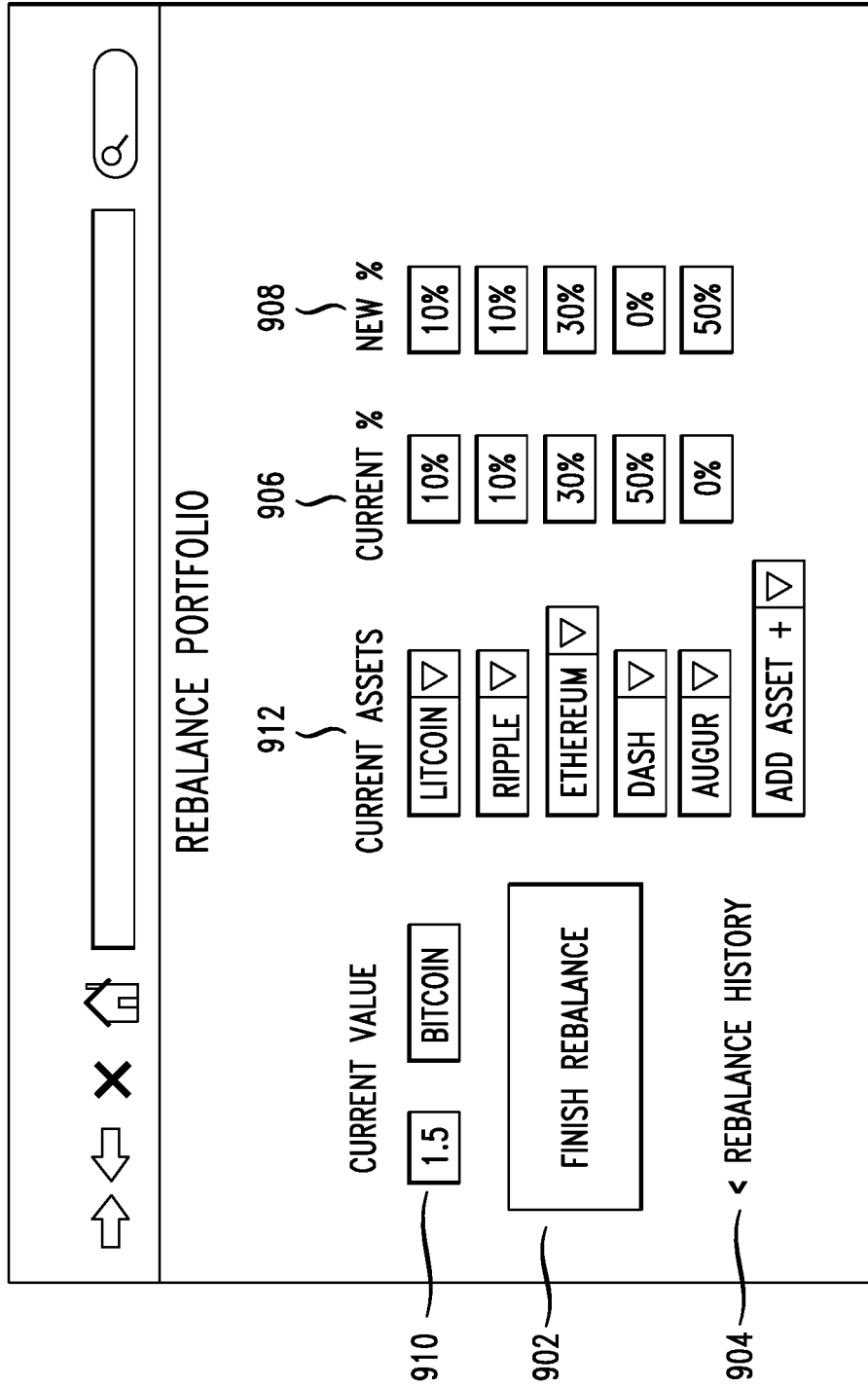
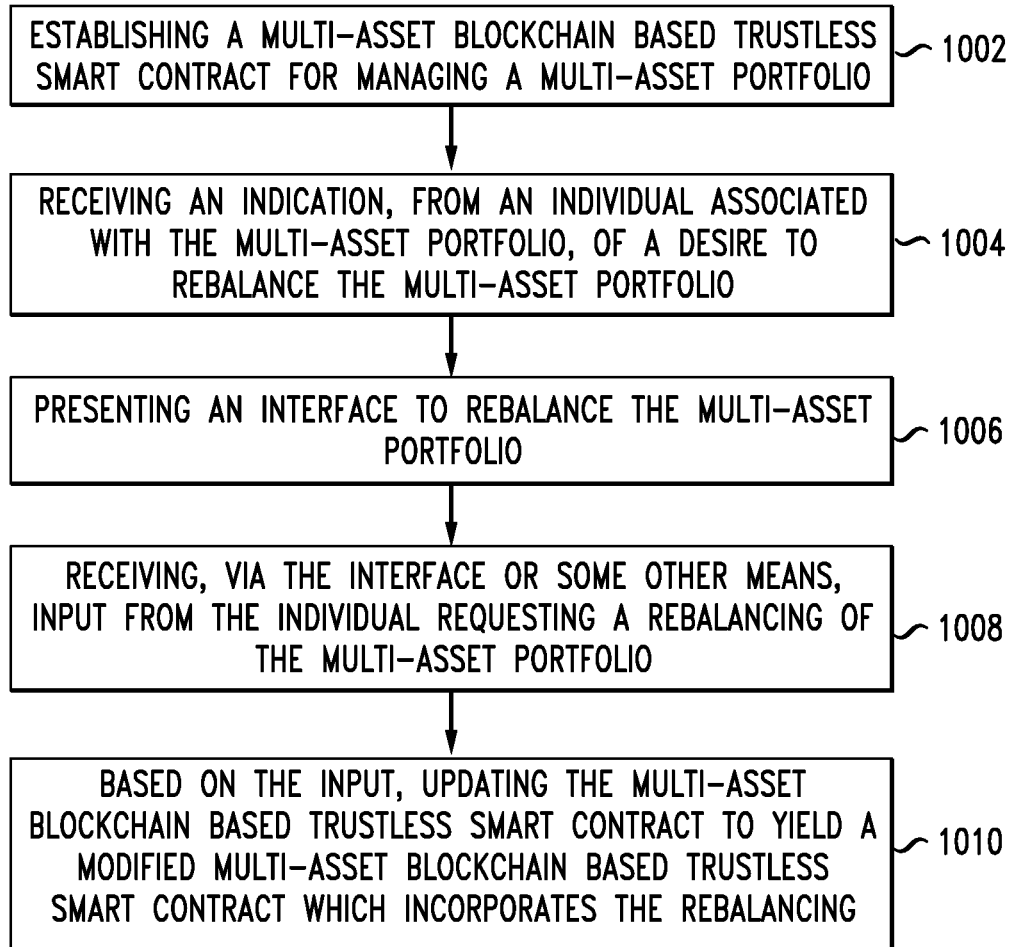

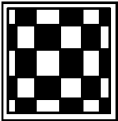
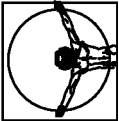
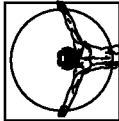


FIG. 9

*FIG. 10*

1100



| | Tweets & Replies | Photos & Videos |
|--|---|-------------------|
| Tweets | | |
|  | Erik Retweeted | |
|  | Lisa @Lisa - 4h Creating a new token is really easy on our mobile wallet. You can even buy XCP within the wallet via@ShapeShift_io @ErikVoorhees | |
| | ↳ 3 ♥ 1 ⋮ | View conversation |
|  | Erik @Erik - 4h My portfolio was at the top of the Leader Boards this week! Nailed the DASH doubling. Find it, here: http://EXAMPLE.COM | |
| | ↳ 1 ♥ 4 ⋮ | View conversation |
|  | Erik @Erik - 4h Cryptocurrency markets been cray cray. View, buy or sell all the top coins direct at coincap.io #bitcoin #ethereum #lifecoin | |
| | ↳ 5 ♥ 10 ⋮ | |

1102

FIG. 11

1200 ↙

↔
✕
🏠

🔍

SUMMARY OF PORTFOLIO FOR erik@shapeshift.io

| CURRENT VALUE | INITIAL VALUE | ASSETS | PERCENT | CURRENT RATE (/BTC) | GAIN/LOSS % |
|---------------|---------------|----------|---------|---------------------|-------------|
| 150 ETH | 100 ETH | LITCOIN | 10% | 12.633 LTC | -10% |
| \$672.00 USD | \$408.00 USD | RIPPLE | 10% | 7471.88 XRP | +10% |
| 1 ETH | 1 ETH | ETHEREUM | 30% | 787.11 ETH | +0% |
| 2 DASH | 1 DASH | DASH | 50% | 95.94 DASH | +100% |

CLONE PORTFOLIO

 AUTOMATICALLY REBALANCE PORTFOLIO WHEN OWNER DOES

FIG. 12

1300



↩
↪
✕
🏠

🔍

LEADER BOARDS FOR PUBLIC PORTFOLIOS

| OWNER | PORTFOLIO NAME | TOTAL RETURN | WEEK | MONTH | YEAR |
|--------|------------------------------|--------------|------|-------|------|
| USER A | MY TOP FIVE CRYPTOS | 50% | 50% | NO | NO |
| USER B | MY TOP 20 CRYPTOS | 250% | 42% | 106% | NO |
| USER C | MOST MarketCap CRYPTOS | 550% | 39% | 145% | 350% |
| USER D | ETF MarketCap WEIGHT TOP 20 | 246% | 31% | 85% | 241% |
| USER D | ETF INVERSE MarketCap TOP 20 | 110% | 22% | 96% | NO |
| USER E | ACTIVELY TRADED - SHORT TERM | 150% | 17% | 52% | 75% |
| USER E | ACTIVELY TRADED - LONG TERM | 249% | 6% | 82% | 45% |

CLICK TO CLONE AND FOLLOW THE PORTFOLIO

1304

1306

1308

1310

1312

1302

FIG. 13

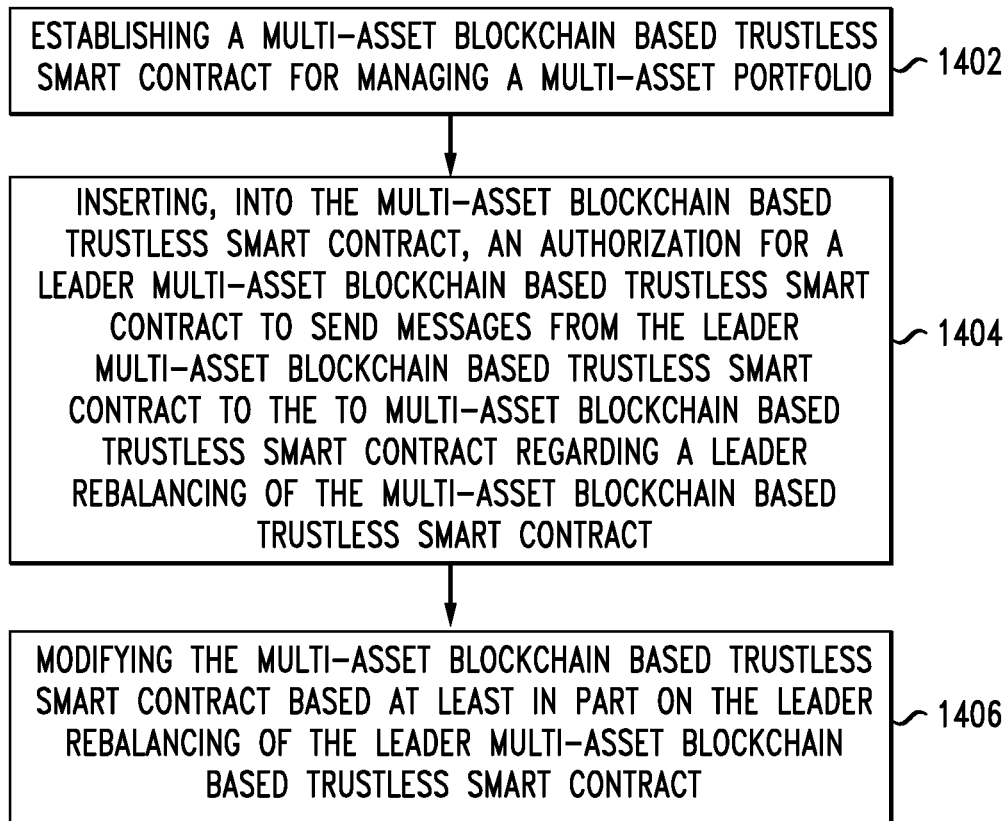


FIG. 14

1500 ↙

↔
✕
🏠

🔍

SELL YOUR PORTFOLIO

1512

1512

1510

1512

1512

1512

PORTFOLIO SUMMARY

| CURRENT VALUE | INITIAL VALUE | ASSETS | PERCENT | CURRENT RATE (/BTC) | GAIN/LOSS % |
|---------------|---------------|----------|---------|---------------------|-------------|
| 1.50 BITCOIN | 1.0 BITCOIN | LITCOIN | 10% | 12.633 LTC | -10% |
| \$672.00 USD | \$408.00 USD | RIPPLE | 10% | 7471.88 XRP | +10% |
| 1.5 ETH | 1.5 ETH | ETHEREUM | 30% | 787.11 ETH | +0% |
| 2.0 DASH | 1.0 DASH | DASH | 50% | 95.94 DASH | +100% |

1502

LIQUIDATION DETAILS

SEND 1.5 ETH TO:

1pJmV3qfvL7SuYo34YihAf5sRCW4qSinyC

LIQUIDATE PORTFOLIO

1504

1506

1508

FIG. 15

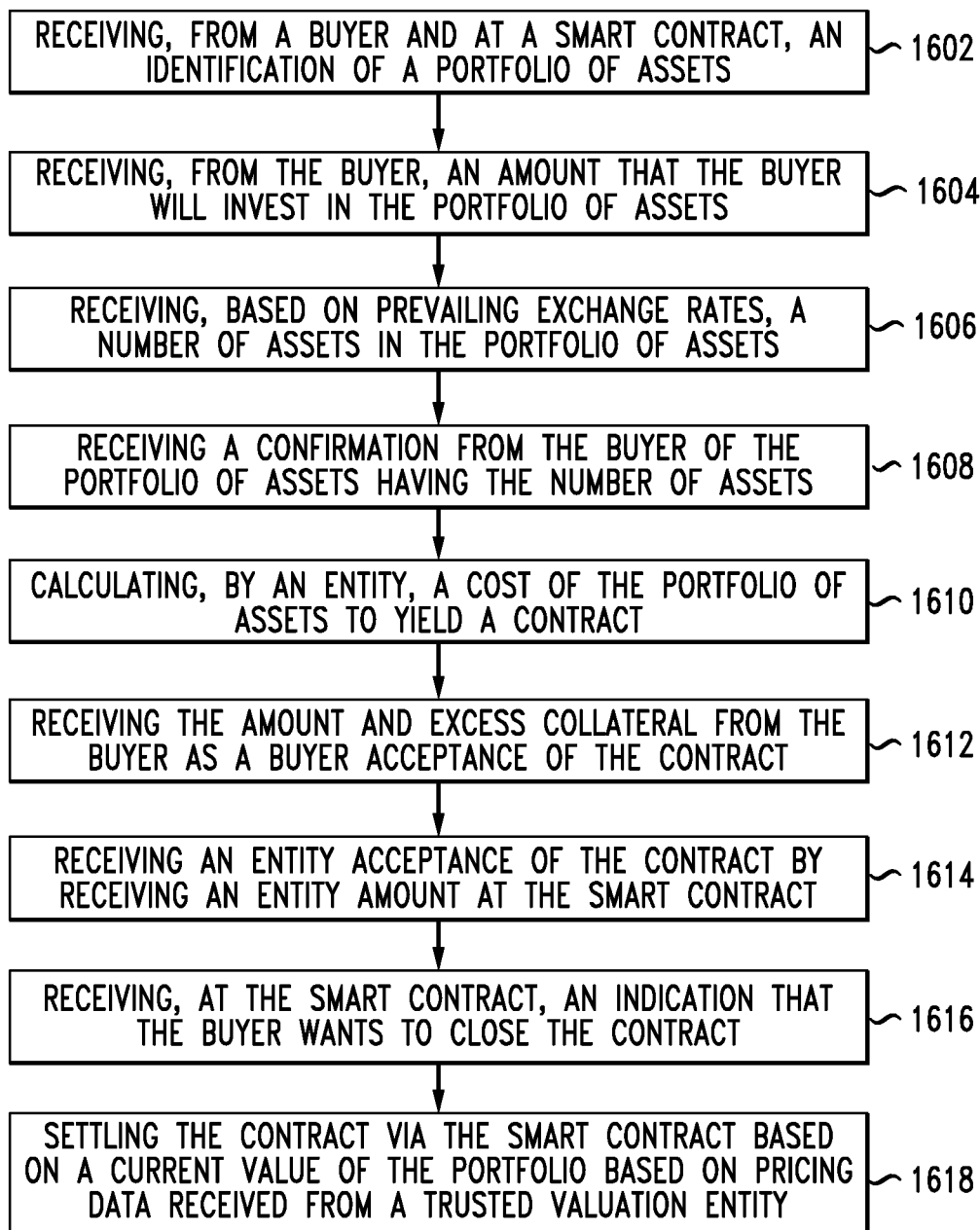


FIG. 16A

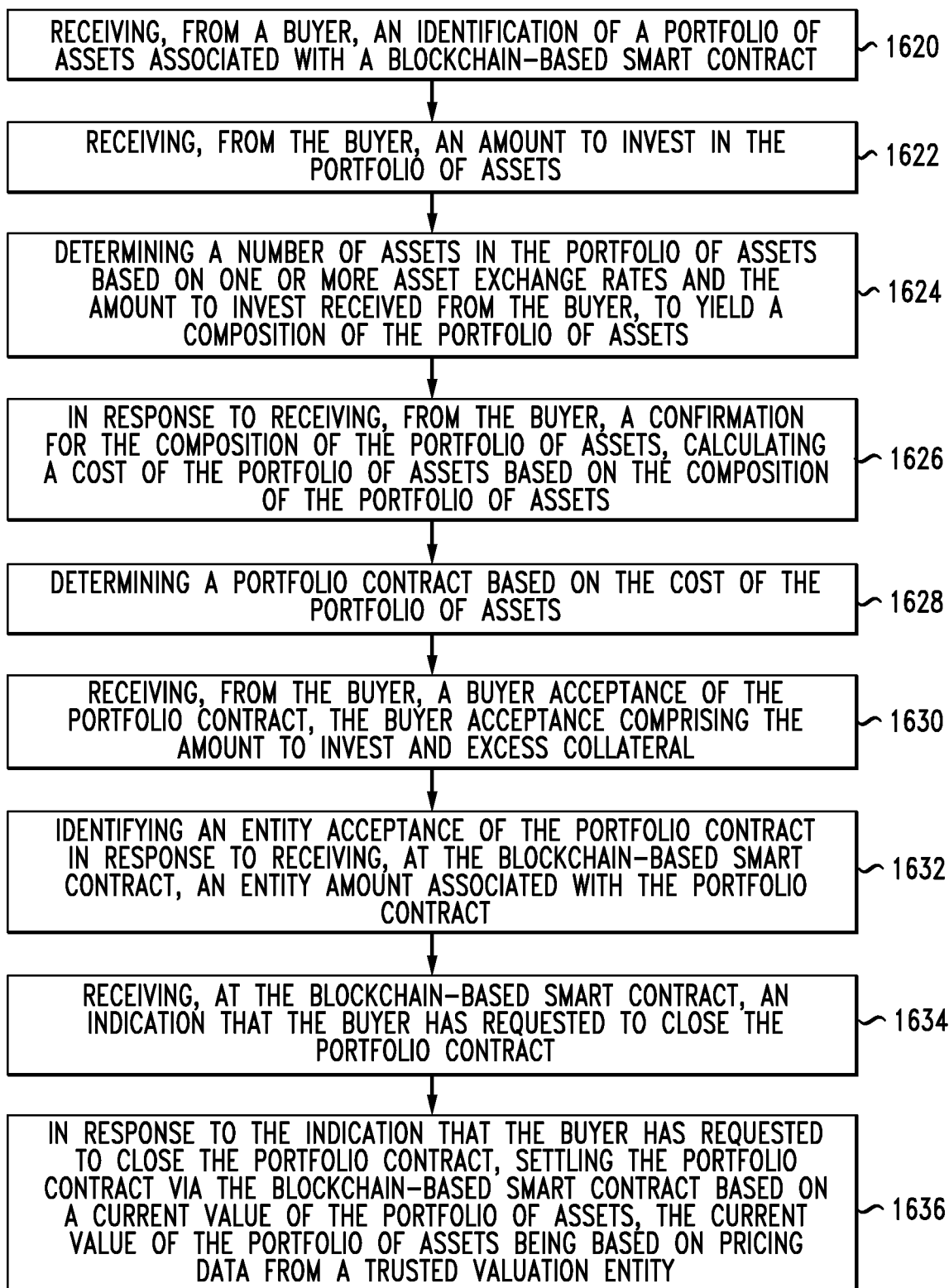


FIG. 16B

SYSTEM AND METHOD OF PROVIDING A CONTRACT-CREATOR APPLICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of, and priority to, U.S. Nonprovisional patent application Ser. No. 15/715,746, filed on Sep. 26, 2017, which claims priority benefit to U.S. Provisional Patent Application No. 62/399,763, filed on Sep. 26, 2016, U.S. Provisional Patent Application No. 62/453,416, filed on Feb. 1, 2017, U.S. Provisional Patent Application No. 62/453,350, filed on Feb. 1, 2017, U.S. Provisional Patent Application No. 62/453,379, filed on Feb. 1, 2017, U.S. Provisional Patent Application No. 62/453,384, filed on Feb. 1, 2017, the entire contents of each of which are herein incorporated by reference in their entireties.

TECHNICAL FIELD

[0002] The present disclosure relates to providing a contract creator for management of multi-asset block-chain based portfolios.

BACKGROUND

[0003] In traditional approaches to asset management, assets are typically held by entities in a custodial manner. This creates a risk to the buyer, as the buyer's assets are subject to any errors or improprieties by the entities in custody of the assets, such as fraud, forged data, deleted data, and so forth. An enormous legal and regulatory structure exists to prevent, detect, and dissuade such errors by setting forth processes and rules for record keeping of company data, such as documents, emails, transactions, contracts, etc., and setting forth specific disclosure and fiduciary requirements.

[0004] Financial institutions are also subject to frequent audits designed to ensure those records and processes accurately represent the firm's assets, liabilities, and operations. Despite the frequent audits and the legal and regulatory structure that exists, fraud and insolvency cases occur across the entire landscape of asset management with surprising regularity. The continual failure to prevent these issues highlights the risks inherent in using third parties to manage assets. Current technologies similarly fail to prevent these issues.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The disclosure will be readily understood by the following detailed description in conjunction with the accompanying drawings in which:

[0006] FIG. 1 illustrates example computing components of a computing device according to one or more aspects of this disclosure;

[0007] FIG. 2 illustrates an example process for portfolio swaps based on smart contracts and blockchain technologies;

[0008] FIG. 3 illustrates an example graphical interface for building a portfolio of assets;

[0009] FIG. 4 illustrates an example graphical interface for buying a portfolio of assets;

[0010] FIG. 5 illustrates an example graphical interface for completing payment for a portfolio of assets;

[0011] FIG. 6 illustrates an example graphical interface for viewing the current composition, status, and value of a portfolio of assets selecting portfolio actions;

[0012] FIG. 7 illustrates a method aspect for a smart contract creator;

[0013] FIG. 8 illustrates a method aspect for a multi-validator oracle;

[0014] FIG. 9 illustrates an example graphical interface for rebalancing a portfolio of assets;

[0015] FIG. 10 illustrates an example method aspect for a rebalancing approach;

[0016] FIG. 11 illustrates an example graphical interface for sharing a portfolio of assets with users on a network;

[0017] FIG. 12 illustrates an example graphical interface for following a portfolio of assets;

[0018] FIG. 13 illustrates an example graphical interface for providing leader boards based on various portfolios of assets;

[0019] FIG. 14 illustrates a method for enabling a leader/follower asset management approach;

[0020] FIG. 15 illustrates an example graphical interface for liquidating a portfolio of assets; and

[0021] FIGS. 16A and 16B illustrate example methods of managing a multi-asset portfolio.

DESCRIPTION OF EXAMPLE EMBODIMENTS

Overview

[0022] Blockchain technology can reduce or eliminate the risks in asset management. Blockchain technologies depend on the existence of various networks such as the Internet, and allow users to exchange assets such as digital currency using a decentralized verification system implemented and deployed over multiple servers. The process for buying, securing/storing and vetting blockchain assets has many technical steps and challenges. For example, any investor who wants to build a portfolio of blockchain assets should (1) conduct in-depth research and analysis into which blockchain assets to buy, (2) research and register at multiple asset exchanges to purchase blockchain assets, (3) securely generate and store the private keys for each blockchain asset the investor wants to purchase, and (4) manually buy or sell assets when they want to change the composition of the portfolio.

[0023] Previous solutions for trading assets allow users to gain exposure to thousands of available tickers (like stocks, bonds, currencies, etc.). Such applications are designed for tickers to be traded on a "single asset basis". That is, each and every swap is for a single asset (i.e. Apple for IBM, or USD for euros, etc.). The bitcoin blockchain system can be used for blockchain assets and cryptocurrencies. However, the bitcoin blockchain system requires a wallet for every asset. The wallet stores the user's private key and address for that asset. For every different type of cryptocurrency, a user must download the respective wallet that stores a private key and an address for that user. This requirement for a separate wallet, private key, etc. for each type of blockchain asset is cumbersome, and creates significant technical limitations and burdens pertaining to data storage, bandwidth, digital security, computing efficiency, software and data compatibility, etc.

[0024] The disclosed technologies provide a technical solution to address the limitations and burdens pertaining to data storage, bandwidth, digital security, computing effi-

ciency, software and data compatibility, and provides various technical advantages which eliminate or reduce asset management risks and other limitations. For example, the concepts disclosed herein can reduce or eliminate such risks, as well as other risks such as the risk of custodianship in the blockchain asset trading world and beyond, including the broader financial services and investment management industries.

[0025] Disclosed herein are technologies and applications for blockchain-based smart contract driven asset management. The approaches disclosed herein allow users to: (1) Build a portfolio of blockchain assets from a user-friendly interface, (2) Purchase an entire portfolio in a single transaction (i.e., rather than a single asset basis transaction, transactions can involve a plurality of assets being acquired in a single step), (3) Secure the entire portfolio under a single private key, which only the investor controls, from beginning to end (i.e. no custodial or counterparty risk), (4) Manage and maintain multiple blockchain assets and asset types without storing and managing a different wallet for each type of the blockchain assets, (5) Rebalance the portfolio as desired through a single signed transaction, (6) Leverage the collective knowledge of the crowd to “socially” design a crypto portfolio, (7) Manage a portfolio of assets and “follow” top blockchain asset portfolio managers and traders without giving up custody of their assets—a new type of asset management process), and (8) Store, trade, and manage an entire group of assets using a single private key.

[0026] Blockchain technologies provide significant improvements over traditional asset management and data recording procedures. Blockchain represent an evolution in web and database technology. Leveraging blockchain technology enables new services and service improvements not previously feasible or even possible. For example, the approaches herein can implement blockchain technologies to provide a new service and technical environment for portfolio management with reduced/eliminated custodial risk, such as the leader/follower concepts further described herein, which allows a leader to direct the follower’s investments without taking custody of the assets.

[0027] The approaches set forth herein can provide secure and efficient technologies and procedures for managing a portfolio of blockchain assets and implementing various asset management functionalities such as portfolio rebalancing. The approaches herein can implement a distributed architecture for hosting a blockchain and distributed information which together can enhance asset and portfolio information, asset and information security, and user control and flexibility. The process outlined herein can also reduce costs.

[0028] With this technology, users can own and manage portfolios of blockchain (or any digital) assets using the blockchain, smart contracts, and related infrastructure disclosed herein. Users do not need to keep coins on an exchange or even interface with an exchange. Users do not need to download the wallets/clients of any specific blockchain asset which they want to purchase and store. Instead, users can own and manage portfolios of blockchain assets using the blockchain, smart contracts, and related infrastructure disclosed herein.

[0029] While the description herein focuses on blockchain assets, it should be noted that the technologies herein can also be used to manage non-blockchain assets and build

portfolios with non-blockchain assets, such as stocks, bonds, real estate, contracts, leases, and other types of assets. The technologies herein can be used to build portfolios of any assets having pricing data-feeds. The benefits and use cases for building non-blockchain asset portfolios can vary from those associated with building blockchain asset portfolios. This is due at least in part to the differences in the infrastructure and industry associated with these types of assets.

[0030] The approaches disclosed herein provide significant advantages of transparency and auditability. In the present disclosure, “the contract is the code,” which means that using blockchain technology and smart contracts, everything is transparent and secured by cryptography. Individuals cannot forge, erase, or add data inappropriately. The system provides complete transparency in view of who owns the assets and their providence over time. This new system provides a new infrastructure with greater cyber security and overall security, and eliminates the need to trust others and rely on third parties to be honest. One non-limiting example of a platform for developing and deploying smart contracts on the blockchain is the Ethereum Virtual Machine.

[0031] The computer code or code referenced herein represents the intent of parties to the smart contract. The smart contract as implemented herein reduces the amount of interpretation with respect to the contract by establishing that the code is a literal manifestation of the intent of the parties, thus avoiding ambiguity.

[0032] A multi-asset swap option disclosed herein limits or eliminates the risk inherent in the use of financial swap arrangements which require trusting one or more parties to deliver or otherwise satisfy obligations upon settlement, known as performance. Every swap has counterparty risk, and this risk will be priced and paid ultimately by one or more parties in the arrangement. Not having mathematical certainty of performance in the swap contract is an unnecessary loss to all involved in the contract.

[0033] The problems outlined above, among others, are addressed by the concepts disclosed herein. For example, the approaches disclosed herein can implement a blockchain-based smart contract paired with an oracle data feed in which the performance and settlement of the swap arrangement occurs autonomously and without risk to either party. Both parties, when they enter in the contract, know with mathematical certainty that the other side does not need to perform anything. With the approaches herein, there is no counterparty risk and no need for a clearing house. The blockchain-based smart contract creates an auditable, transparent trail and the code executes the intent of the parties. The use of the smart contract herein reduces the likelihood of failure to perform, eliminates non-performance risks, and reduces the cost of entering, managing, and executing a swap agreement.

[0034] The use of the smart contract herein can also reduce the cost of dealing with such failures to perform and eliminates the need for a central clearing house. In the derivatives world, there are over-the-counter (OTC) derivatives which can be an agreement directly between the two parties. In that case, the parties have counterparty risk should one or the other party not fulfill their obligations. In the case where both parties employ the services of a clearing house as a trusted third party, there is always a risk that the trusted third party may not manage and enforce the swap. The approaches herein eliminate the risks in both scenarios above, as well as their associated costs. Again, in the

disclosed solution, there is no need for a clearing house, and in the OTC scenario, both parties do not have trust each other thus eliminating counterparty risk.

[0035] Non-limiting examples and aspects of the technologies disclosed herein include, without limitation, a trustless blockchain multi-asset ‘swaption’ mechanism, a multi-asset trustless swap rebalance mechanism, a trustless multi-asset leader-follower mechanism, a contract creator mechanism and a multi-validator oracle. One or more of these individual examples or features can be combined and/or otherwise function together with other examples or features disclosed herein, to form a specific application or configuration. For example, one or more of the components disclosed herein can be combined together to create a portfolio index swap marketplace (herein called a “PRISM”). “PRISM” is a working product name and refers to an application and service that enables buying and selling of multi-asset portfolios based on blockchain technology and smart contracts.

[0036] A number of examples will be provided related to blockchain assets or cryptocurrencies, such as Bitcoins, Ethereum, and other digital currencies and assets. However, it is noted that the concepts herein apply to portfolios of any type of blockchain asset, digital currency or asset, and/or non-blockchain assets such as stock equities, bonds, commodities, real estate, contracts, currencies (e.g., dollars), etc.

[0037] As previously noted, users do not have to keep assets on an exchange, or interface with an exchange, third party or counterparty. This eliminates significant risks such as fraud, non-performance, breach of duty, theft, etc. Moreover, users do not have to download the specific wallets or clients for respective blockchain assets that users want to purchase and store securely. Instead, users can own and manage one or more portfolios of assets, each of which can include a variety of assets such as blockchain assets and/or “traditional assets” (e.g., equities, bonds, etc.) using a blockchain and associated security features like public/private key infrastructure. Each individual portfolio is secured using the public/private key infrastructure and transactions are verified using the blockchain.

[0038] In this respect, the present technologies provide a new computing infrastructure, environment, and procedure for managing digital assets and multi-asset portfolios, with significant improvements over existing technologies and procedures, including technologies and procedures for data storage, security, management, communication, efficiency, etc. The new computing infrastructure, environment, and procedure provide distributed data storage, enhancement, security, control, processing, verification, etc., and enhance the information and functionality of data and systems across the distributed environment. The new process disclosed herein allows value creators (the portfolio managers or traders) to connect directly with consumers (those users who need their assets managed professionally) without numerous intermediaries like clearing houses, custodians, banks, etc. This is accomplished at least partly by collapsing the infrastructure costs of such intermediaries onto the blockchain and associated technologies (smart contracts, PKI infrastructure, modern cryptography, etc.).

[0039] Blockchain technologies are designed for trading single assets and are incapable of performing multi asset transactions or managing portfolios of different types of assets, such as blockchain assets and non-blockchain assets (e.g., real-world assets). However, the technologies dis-

closed herein address these limitations and provide various other improvements to blockchain and non-blockchain technologies, including flexibility, security, and performance improvements. For example, the technologies provide a multi-asset solution which allows portfolios to be built with different types of assets and enables swaps of entire portfolios as opposed to merely single asset transactions.

[0040] The disclosed technologies also implement a smart contract based oracle through which the smart contract can receive and validate pricing data, including multi-sig, multi-validation, or fully decentralized.

[0041] in a multi-signature context, rather than using a single private key to authorize the transaction, the system uses multiple private keys to sign the transaction together before the transaction happens. The system can require a number of keys which can vary in different configurations, such as 2 out of 3 possible private keys to verify the transaction or 99 out of 100, or 3 out of 8, and so forth. The multi-signature arrangement removes a point of failure from the process and provides a safer transaction. This can apply to the oracle disclosed herein and/or to other components.

[0042] The present technologies also provide a unique leader-follower functionality along with various benefits including mitigation of custodial risk. Unlike current solutions, the present technologies also provide the architecture and functionality for enabling in-contract rebalancing of a portfolio of assets, which can be performed for the entire portfolio without opening and closing every single asset swap every time the user wants to rebalance the portfolio. The technologies also provide an architecture and environment that integrates a social networking component. For example, with the disclosed technologies, users can share their portfolio with other users on one or more networks, such as a social network. Users can also view and select portfolios of other users to follow selected portfolio(s) from the other users.

[0043] The disclosed technologies can be deployed using different platforms and environments, including desktop applications, browser applications, etc., and are not limited to a specific platform. The present technologies also provide an array of smart contract security features, such as the ability to “pause contracts”, “manually override contracts”, etc., as well as a “training wheels” method as further described herein. In some aspects of the present disclosure, only the buyer can initiate settlement, which can prevent settlements being triggered without the buyer’s consent, such as based on margin calls, the expiration of time, or by the seller. The buyer has the option to settle a swap at any time and the seller can be prevented from triggering a settlement. This can provide significant security and usability implications.

[0044] The technologies disclosed herein also provide a new architecture and code-base which allows for a unique “leader and follower utility.” This functionality allows a “leader” to invest/allocate digital assets corresponding to “followers”, without taking custody of those assets. This feature and functionality eliminates typical custodial arrangements and their associated risks. With the disclosed technologies, a third party does not have to take custody of client assets. Instead, users retain possession of their assets, which can be aggregated into “portfolios” stored and/or verified using a blockchain network. Portfolios can include various types of blockchain assets (e.g., cryptocurrencies, etc.), as well as other non-blockchain assets.

[0045] A uniquely implemented blockchain-based oracle, which can provide current pricing information and can be used as a data feed in the present application. This oracle and its configuration in the present application can include a number of security features, such as a “manual override”, “contract-pause”, “training wheels” processes, and others.

[0046] The present disclosure includes systems, methods, and computer-readable storage devices for implementing blockchain-based smart contracts for asset management, including trustless blockchain swaption contracts of multi-asset portfolios for example. In some examples, a system or method can include receiving, from a buyer and at a smart contract, an identification of a portfolio of blockchain assets (e.g., digital currencies such as Bitcoin or Ethereum), receiving, from the buyer, an amount that the buyer will invest in the portfolio of blockchain assets and receiving, based on prevailing exchange rates, a number of blockchain assets in the portfolio.

[0047] The system or method further includes receiving a confirmation from the buyer of the portfolio having the number of blockchain assets, calculating, by an entity, a cost of the portfolio of blockchain assets to yield a contract, receiving the amount and excess collateral from the buyer as a buyer acceptance of the contract, and receiving an entity acceptance of the contract by receiving an amount from the seller or other entity (i.e., an asset value amount such as a value in dollars or value of cryptocurrency) at the smart contract. Private keys of a sending address associated with the entity can be used as an entity signature key for the contract.

[0048] The system or method further includes receiving, at the smart contract, an indication that the buyer wants to close the contract. This indication can be associated with a signed message from a buyer private key. The system or method can also include settling the arrangement via the smart contract based on a current value of the portfolio, which can be calculated based on pricing data received from a trusted valuation entity, such as the oracle or another source.

[0049] Also disclosed is an approach for rebalancing multi-asset smart contracts. In some examples, a system or method includes establishing a multi-asset blockchain-based trustless smart contract for managing a multi-asset portfolio, receiving an indication, from an individual associated with the multi-asset portfolio, of a desire to rebalance the multi-asset portfolio, presenting the individual with an interface to rebalance the multi-asset portfolio, receiving, through the interface, input from the individual including a rebalancing of the multi-asset portfolio and updating the multi-asset blockchain-based trustless smart contract to yield a modified multi-asset blockchain-based trustless smart contract which incorporates the rebalancing.

[0050] The smart contract can be the same contract, existing in the same place on the blockchain but with a new “state”. The new state reflects the new composition of the portfolio which is recorded in the blockchain. The blockchain is the transparent and relatively immutable record of state changes which the contract undergoes. The system writes a new entry in the blockchain that reflects the new portfolio composition. In another aspect, the system can create a new contract but with different parameters (asset allocations). The old contract can be discarded, deleted or made inactive.

[0051] The indication of the desire to rebalance the multi-asset portfolio can include a request to rebalance one or more assets in the multi-asset portfolio or add/delete one or more assets in the multi-asset portfolio.

Description

[0052] The present disclosure addresses the various issues and limitations in the art outlined above, such as security limitations, performance limitations, inefficiencies, control limitations, flexibility limitations, lack of transparency, infrastructure limitations, etc. Example features and functionalities of the present disclosure include trustless blockchain multi-asset swaption mechanisms, trustless multi-asset rebalance mechanisms, trustless multi-asset leader-follower mechanisms, smart-contract creator mechanisms, multi-validator oracle systems, etc. These example features and functionalities can be implemented in a distributed network and computing environment which can provide various advantages in security, storage, efficiency, bandwidth, computation, control, flexibility, performance, etc., and can support a variety of software application and computing platforms. One or more of these individual components can be combined and work together with any other one or more of the components in various configurations.

[0053] The disclosure first turns to FIG. 1 which illustrates example hardware components for a computing system that can be implemented for various aspects of the present disclosure. The disclosure will then turn to a description of example multi-asset management mechanisms, architectures, and concepts.

[0054] With reference to FIG. 1, an exemplary system and/or computing device **100** includes a processing unit (CPU or processor) **110** and a system bus **105** that couples various system components including the system memory **115** such as read only memory (ROM) **120** and random access memory (RAM) **125** to the processor **110**. The system **100** can include a cache **112** of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor **110**. The system **100** copies data from the memory **115**, **120**, and/or **125** and/or the storage device **130** to the cache **112** for quick access by the processor **110**. In this way, the cache provides a performance boost that avoids processor **110** delays while waiting for data. These and other modules can control or be configured to control the processor **110** to perform various operations or actions. Other system memory **115** may be available for use as well. The memory **115** can include multiple different types of memory with different performance characteristics. It can be appreciated that the disclosure may operate on a computing device **100** with more than one processor **110** or on a group or cluster of computing devices networked together to provide greater processing capability. The processor **110** can include any general purpose processor and a hardware module or software module, such as module **1** **132**, module **2** **134**, and module **3** **136** stored in storage device **130**, configured to control the processor **110** as well as a special-purpose processor where software instructions are incorporated into the processor. The processor **110** may be a self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric. The processor **110** can include multiple processors, such as a system having multiple, physically separate processors in

different sockets, or a system having multiple processor cores on a single physical chip. Similarly, the processor **110** can include multiple distributed processors located in multiple separate computing devices, but working together such as via a communications network. Multiple processors or processor cores can share resources such as memory **115** or the cache **112**, or can operate using independent resources. The processor **110** can include one or more of a state machine, an application specific integrated circuit (ASIC), or a programmable gate array (PGA) including a field PGA.

[0055] The system bus **105** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. A basic input/output system (BIOS) stored in ROM **120** or the like, may provide the basic routine that helps to transfer information between elements within the computing device **100**, such as during start-up. The computing device **100** further includes storage devices **130** or computer-readable storage media such as a hard disk drive, a magnetic disk drive, an optical disk drive, tape drive, solid-state drive, RAM drive, removable storage devices, a redundant array of inexpensive disks (RAID), hybrid storage device, or the like. The storage device **130** is connected to the system bus **105** by a drive interface. The drives and the associated computer-readable storage devices provide non-volatile storage of computer-readable instructions, data structures, program modules and other data for the computing device **100**. In one aspect, a hardware module that performs a particular function includes the software component stored in a tangible computer-readable storage device in connection with the necessary hardware components, such as the processor **110**, bus **105**, an output device such as a display **135**, and so forth, to carry out a particular function. In another aspect, the system can use a processor and computer-readable storage device to store instructions which, when executed by the processor, cause the processor to perform operations, a method or other specific actions. The basic components and appropriate variations can be modified depending on the type of device, such as whether the computing device **100** is a small, handheld computing device, a desktop computer, or a computer server. When the processor **110** executes instructions to perform “operations”, the processor **110** can perform the operations directly and/or facilitate, direct, or cooperate with another device or component to perform the operations.

[0056] Although the exemplary embodiment(s) described herein employs a storage device such as a hard disk **130**, other types of computer-readable storage devices which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, digital versatile disks (DVDs), cartridges, random access memories (RAMs) **125**, read only memory (ROM) **120**, a cable containing a bit stream and the like, may also be used in the exemplary operating environment. According to this disclosure, tangible computer-readable storage media, computer-readable storage devices, computer-readable storage media, and computer-readable memory devices, expressly exclude media such as transitory waves, energy, carrier signals, electromagnetic waves, and signals per se.

[0057] To enable user interaction with the computing device **100**, an input device **145** represents any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output

device **135** can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems enable a user to provide multiple types of input to communicate with the computing device **100**. The communications interface **140** generally governs and manages the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic hardware depicted may easily be substituted for improved hardware or firmware arrangements as they are developed.

[0058] For clarity of explanation, the illustrative system embodiment is presented as including individual functional blocks including functional blocks labeled as a “processor” or processor **110**. The functions these blocks represent may be provided through the use of either shared or dedicated hardware, including, but not limited to, hardware capable of executing software and hardware, such as a processor **110**, that is purpose-built to operate as an equivalent to software executing on a general purpose processor. For example the functions of one or more processors presented in FIG. **1** can be provided by a single shared processor or multiple processors. (Use of the term “processor” should not be construed to refer exclusively to hardware capable of executing software.) Illustrative embodiments may include microprocessor and/or digital signal processor (DSP) hardware, read-only memory (ROM) **120** for storing software performing the operations described below, and random access memory (RAM) **125** for storing results. Very large scale integration (VLSI) hardware embodiments, as well as custom VLSI circuitry in combination with a general purpose DSP circuit, may also be provided.

[0059] The logical operations of the various embodiments are implemented as: (1) a sequence of computer implemented steps, operations, or procedures running on a programmable circuit within a general use computer; (2) a sequence of computer implemented steps, operations, or procedures running on a specific-use programmable circuit; and/or (3) interconnected machine modules or program engines within the programmable circuits. The system **100** shown in FIG. **1** can practice all or part of the recited methods, can be a part of the recited systems, and/or can operate according to instructions in the recited tangible computer-readable storage devices. Such logical operations can be implemented as modules configured to control the processor **110** to perform particular functions according to the programming of the module. For example, FIG. **1** illustrates three modules Mod1 **132**, Mod2 **134** and Mod3 **136** which are modules configured to control the processor **110**. These modules may be stored on the storage device **130** and loaded into RAM **125** or memory **115** at runtime or may be stored in other computer-readable memory locations.

[0060] One or more parts of the example computing device **100**, up to and including the entire computing device **100**, can be virtualized. For example, a virtual processor can be a software object that executes according to a particular instruction set, even when a physical processor of the same type as the virtual processor is unavailable. A virtualization layer or a virtual “host” can enable virtualized components of one or more different computing devices or device types by translating virtualized operations to actual operations. Ultimately however, virtualized hardware of every type is implemented or executed by some underlying physical hardware. Thus, a virtualization compute layer can operate on top of a physical compute layer. The virtualization compute

layer can include one or more of a virtual machine, an overlay network, a hypervisor, virtual switching, and any other virtualization application.

[0061] The processor **110** can include all types of processors disclosed herein, including a virtual processor. However, when referring to a virtual processor, the processor **110** includes the software components associated with executing the virtual processor in a virtualization layer and underlying hardware necessary to execute the virtualization layer. The system **100** can include a physical or virtual processor **110** that receive instructions stored in a computer-readable storage device, which cause the processor **110** to perform certain operations. When referring to a virtual processor **110**, the system also includes the underlying physical hardware executing the virtual processor **110**.

[0062] The disclosure now turns to a description of example multi-asset management mechanisms, architectures, and related concepts in accordance with various aspects of the disclosed technologies.

[0063] FIG. 2 illustrates an example application and service **200** for managing the buying and selling of asset groups. In this example, the application uses financial swaps, implemented via a smart contract. The swap can be “secured” and “enforced” through smart contracts. In some examples, the smart contract can be written in one or more computing languages, such as Solidity, the Turing Complete language of Ethereum (or any other blockchain-operable language that would result in the same or similar functionality as disclosed herein). Swaps can be a promise or agreement for each party (the Buyer and the Seller) to swap asset exposures for some amount of time. With swaps, the underlying assets are never exchanged—all that is exchanged is a contract that binds each party to pay one another based on the change in value of the underlying assets as recorded by an agreed upon external data, such as the oracle **210**.

[0064] The application can run smart contracts, which can include autonomous applications or code that run as programmed without downtime, censorship, fraud or third party interference. The application can run on a custom built blockchain architecture, which provides a powerful shared global infrastructure that can move values around and represent ownership of property. This infrastructure enables developers manage records, transactions, and values in a distributed fashion according to instructions, all without a middle man or counterparty and custodial risk. For clarity and explanation purposes, the example application will be described herein with reference to a smart contract on a platform like Ethereum.

[0065] Financial swaps like index swaps and swaptions are financial engineering tools for both trading and risk management. Billions of dollars of these types of swaps are traded daily through the traditional banking system. However, the present disclosure creates an entirely new process for creating, securing, and delivering these swaps to both consumers and institutional users (buyers or sellers). The following discussion outlines various aspects of the disclosed solution.

[0066] The disclosed approaches provide novel technologies for multi-asset portfolio management. In some aspects, the disclosed approaches can use blockchain technology to create, enforce, and secure swaps without counterparty, custodial, and other risks, and cryptocurrency or blockchain assets to collateralize swaps. The disclosed technologies can

also provide a new type of swap, which can be referred to as “a custom portfolio swap”, and differs significantly from an index swap in various ways.

[0067] For example, an index swap has fixed components which are usually based on a large public index calculated and created by a third party, e.g. the Dow Jones Industrial Index or the S&P 500 index. The buyer of an index swap receives that index and cannot customize it. These indexes represent an “investment strategy”. By contrast, the “custom portfolio swap” disclosed herein, which a user can create using the disclosed application, is not limited to an index with fixed components, and can be customized by the user. Thus, the “custom portfolio swap” can represent the user’s own investment strategies, unique from any single index swap on the market today.

[0068] Index swaps are further limited to certain assets based on the indexes available. However, many assets do not have an index. For example, crypto-currencies and real world assets do not have an index or exchange traded funds (ETF) available. Moreover, index swaps (as well as all swaps) rely on data from trusted third-parties. On the other hand, the data used in a “custom portfolio swap” comes from trustless oracles, which differ from trusted third-parties.

[0069] Other differences between the disclosed swaps and other swaps include that the disclosed swaps will always be “over-collateralized”, as opposed to other swaps which typically require fractional margin to enter a swap (vastly under-collateralized). As a result, the disclosed swaps have no counterparty risk and swaps are immune to insolvency for either party. Further, almost all swaps in their current form are designed to be traded on an institutional level, as opposed to a direct-to-consumer product. By contrast, the disclosed technology provides swaps designed as a consumer product, and include significant differences in architecture in order to support this consumer level design.

[0070] Swaps are a form of financial derivative. The interest rate swap business has 300+trillion dollars in outstanding notional value. In the current financial system, swaps provide limited security through: (1) a legally binding contract between the parties to the swap; (2) collateral, margin, and other requirements imposed by clearing houses and brokers around the globe; and (3) the perceived creditworthiness of the two parties to the swap (this is the source of counterparty risk). The new swap disclosed herein can be “secured” and “enforced” through smart contracts written in a blockchain-operable language.

[0071] In the context of blockchains and cryptocurrencies, smart contracts are: (1) pre-written logic (computer code); (2) stored and replicated on a distributed storage platform (e.g. a blockchain); (3) executed/run by a network of computers (usually the same ones running the blockchain); and (4) can result in ledger updates (cryptocurrency payments, etc.).

[0072] In other words, smart contracts are programs that execute “if this happens then do that” that are run on, and are verified by, many computers to ensure trustworthiness. If blockchains provide distributed trustworthy storage, then smart contracts provide distributed trustworthy calculations. Below are three examples illustrating use of smart contracts in the disclosed technologies. The first example illustrates use or replacement of bank accounts with embedded instructions, the second example illustrates replacement of legalese with computer code and the third example provides an actual smart contract example.

Replacement of Bank Accounts with Embedded Instructions

[0073] Bank accounts typically have specific instructions and parameters that define the account's behavior and preferences. For example, bank accounts maintain a balance which indicates the funds available for that bank account. The bank accounts can have instructions defining basic payment and balance management operations for the account. Suppose a user configures the user's bank account to deduct automatic payments to a particular entity (e.g., landlady, utility companies, etc.) for a fixed amount every month, and send the deducted payments to the particular entity. If the balance of the account falls below the amount of an automatic payment configured, the account would not have sufficient funds to cover the amount for the automatic payment. In this scenario, the automatic payment will typically fail, causing the user to possibly receive a fine for insufficient funds and another workflow to be triggered. There are instructions that the user can have set up for the user's bank account, which can define basic conditions or operations for the bank account.

[0074] However, in the context of bank accounts, this process is managed by the bank or institution associated with the bank account. By contrast, a smart contract runs on a blockchain and is therefore managed by many parties rather than a single entity. The distributed control and management of smart contracts provides many advantages as previously explained, such as, without limitation security, reliability, and transparency. Moreover, as further explained herein, smart contracts provide more granular control and a significantly greater degree of flexibility and functionality. Replacing Legalese with Computer Code

[0075] A smart contract provides code which automates the "if this happens, then do that" part of contracts. Computer code behaves in expected ways and does not have the linguistic nuances of human languages. Code has less potential points of contention or ambiguity. In the case of smart contracts, the computer code is replicated on many computers and distributed or decentralized on a blockchain. The computer code is executed by the various computers, which together agree on the results of the code execution. In some cases, a user can have a paper contract with all the "whereas" clauses that lawyers employ, and a clause that points to a smart contract on a blockchain. The smart contract can provide, for example, "both parties to the smart contract agree to run code x and we will abide by the results of the code."

Example Smart Contract

[0076] Below is an example smart contract illustrating example computer code written for use on the Ethereum blockchain:

```
contract tokenRecipient {function receiveApproval(address_from, uint256_value, address_token, bytes_extraData);}
```

```
contract MyToken{
```

```

[0077] /*Public variables of the token*/
[0078] string public standard="Token 0.1";
[0079] string public name;
[0080] string public symbol;
[0081] uint8 public decimals;
[0082] uint256 public total Supply;
[0083] /*This creates an array with all balances*/
[0084] mapping (address=>uint256) public balanceOf;
```

```

[0085] mapping (address=>mapping
(address=>uint256)) public allowance;
[0086] /*This generates a public event on the block-
chain that will notify clients*/event
[0087] Transfer(address indexed from, address indexed
to, uint256 value);
[0088] /*Initializes contract with initial supply tokens to
the creator of the contract*/
[0089] function MyToken(
[0090] uint256 initial Supply,
[0091] string tokenName,
[0092] uint8 decimalUnits,
[0093] string tokenSymbol)
[0094] ){
[0095] balanceOf[msg.sender]=initial Supply; //
Give the creator all initial tokens
[0096] total Supply=initial Supply; // Update total
supply
[0097] name=tokenName; // Set the name for display
purposes
[0098] symbol=tokenSymbol; // Set the symbol for
display purposes
[0099] decimals=decimalUnits; // Amount of deci-
mals for display purposes
[0100] msg.sender.send(msg.value); // Send back any
ether sent accidentally
[0101] }
[0102] /*Send coins*/
[0103] function transfer(address_to, uint256_value)
[0104] if (balanceOf[msg.sender]<_value) throw; //
Check if the sender has enough
[0105] if (balanceOf[_to]+_value<balanceOf[_to])
throw; // Check for overflows
[0106] balanceOf[msg.sender]-=_value; // Subtract
from the sender
[0107] balanceOf[_to]+=_value; // Add the same to
the recipient
[0108] Transfer(msg.sender, _to, _value); // Notify
anyone listening that this transfer took place
[0109] }
[0110] /*Allow another contract to spend some tokens
in your behalf*/
[0111] function approve(address spender, uint256
value)
[0112] returns (bool success)
[0113] allowance[msg.sender][_spender]=_value;
[0114] return true;
[0115] }
[0116] /*Approve and then communicate the approved
contract in a single tx*/
[0117] function approveAndCall(address_spender,
uint256_value, bytes_extraData)
[0118] returns (bool success) {
[0119] tokenRecipient spender=tokenRecipient(_
spender);
[0120] if (approve(_spender, _value)) {
[0121] spender.receiveApproval(msg.sender,
_value, this, _extraData);
[0122] return true;
[0123] }
[0124] }
[0125] /*A contract attempts to get the coins*/
[0126] function transferFrom(address_from, address_
to, uint256_value) returns (bool success) {
```



```

[0127] if (balanceOf[_from]<_value) throw; // Check
if the sender has enough
[0128] if (balanceOf[_to]+_value<balanceOf[_to])
throw; // Check for overflows
[0129] if (_value>allowance[_from][msg.sender])
throw; // Check allowance
[0130] balanceOf[_from]-=_value; // Subtract from
the sender
[0131] balanceOf[_to]+=_value; // Add the same to
the recipient
[0132] allowance[_from][msg.sender]-=_value;
[0133] Transfer(from, _to, _value);
[0134] return true;
[0135] }
[0136] /*This unnamed function is called whenever
someone tries to send ether to it*/
[0137] function () {
[0138] throw; // Prevents accidental sending of ether
[0139] }
}

```

[0140] The example contract generates 10 thousand tokens to the creator of the contract, and then allows anyone with enough balance to send it to others. These tokens are the minimum tradeable unit and cannot be subdivided, but for the final users could be presented as a 100 units subdividable by 100 subunits, so owning a single token would represent having 0.01% of the total.

[0141] The above example contract provides unique control and flexibilities that automated banking payments lack. For example, a user's bank is the ultimate guardian of the user's bank account. The bank has complete control, and can arbitrarily add money to the account or subtract it. In a blockchain ecosystem, there should be no single source of control. The distributed layout with consensus mechanisms in blockchain allows multiple parties to check and re-check and update the ledgers. Anything in the blockchain that does not conform to pre-agreed rules, can be rejected by any of the participants.

[0142] The above example contract also provides code control and validation as opposed to automated banking accounts. With a bank account, there is some logic creating transactions on a monthly basis. That code resides on a computer and is executed by a party (the bank). The bank has full possession and control. In the banking context, there are no external validations. With smart contracts running on a blockchain, the logic runs in parallel on all participating computers, and the results are compared by all participants. Participants only change their own version of the ledger if they agree with the results. Thus, each participant has code control and validation capabilities.

[0143] In addition, the above example contract provides a level of transparency that is not available in the banking context. For example, for all participants in a blockchain ecosystem to run the same code, each verifying the others, the logic of the smart contract is available and visible to all. This allows any participant to review a smart contract, and decide whether to use the logic after first reviewing the logic. If the participant does not like or approve of the logic, that participant has the option to reject the logic (i.e., not to use it). There can be smart contracts for general usage, as well as specific smart contracts. The transparency is useful to stakeholders of the contract to agree on what happens.

[0144] The above example contract provides significant flexibility to make adjustments and customizations that are

not otherwise available in banking systems. For example, the logic that a user can run within their bank account is limited to recurring payments, and possibly a few other basic options. A user cannot, for example, automate a payment from their salary account to their savings account every day it is sunny, then have it all sent back when there is a storm (the 'saving up for a rainy day' smart contract). By contrast, a "Turing complete" smart contract can do anything that a normal computer can do, including numerous operations and customizations.

[0145] Smart contracts are useful when there are multiple parties who may not trust each other fully, as each party can compare their version of events with each other. For example, when two banks do a complex derivative trade with each other that does not go through a clearing house, it is called an "Over The Counter" or OTC trade. These are agreements between the two banks, without third party validation. These trades are usually bets—i.e. variations of "if this happens before the end of the year then you pay me, else I pay you". Both parties have a copy of the original trade documents (the terms and conditions of the trade), and they both have a view on the external dependencies of the trade. The parties should both therefore agree on the outcome of the trade i.e. who wins the bet. However, parties may not always agree on the outcome of the trade.

[0146] A mismatch or "break" can occur in the dependencies, where parties do not agree on the outcome of the trade, due to a number of things. For example, problems can include: (1) A mutual misunderstanding of the initial trade terms; (2) Confusion due to multiple copies of the original trade terms (usually there is back-and-forth on the wording of the documents, with in-house lawyers on both sides trying to protect their interests); or (3) A disagreement with what actually happened in the external dependencies.

[0147] With a smart contract, there is only one set of trade terms, written in computer code, which is more precise than a contract written in legal terms, and agreed upon up-front. The external dependencies (price of oil, share price of a stock, etc.) can be fed in via a mutually agreed feed. The contract will live on a blockchain, and run when a condition occurs (e.g., when an event happens or the bet expires). The bet payout can be stored in the smart contract itself. The contract can be "primed" by both parties funding the account with their maximum loss, and the payout is made when the condition occurs (e.g., the event). Also, a lot of trades in financial services are currently done on credit and margined or collateralized; the necessity to pre-fund the total payout with the full value of the potential payout, in the currency/asset of the payout may not be attractive for some use cases. In another aspect of the application, concepts such as margin considerations can be implemented as well in institutional cases where collateralization may be too costly.

[0148] A further description of smart contract offerings in accordance with various aspects of the disclosed technologies is provided in the following discussion.

[0149] Blockchains (e.g., Bitcoin, Ethereum, etc.) have varying degrees of effectiveness in running smart contracts. Bitcoin's platform is great for processing bitcoin transactions, but otherwise limited in compute ability. Within the scripts of bitcoin transactions, there is a very limited ability to implement rich logic. An example of what is possible in bitcoin is logic requiring multiple signatories to sign a transaction before a payment is made, like needing two signatories in a check. However, major changes would need

to be made to both the mining functions and the mining incentive schemes to enable smart contracts properly on Bitcoin's blockchain.

[0150] Sidechains (i.e. blockchains connected to Bitcoin's main blockchain) can enable smart contract functionality by having different blockchains running in parallel to Bitcoin. These sidechains can be programmed with an ability to jump value between Bitcoin's main chain and the side chains, such that side chains could be used to execute logic.

[0151] NXT is a public blockchain platform which includes a selection of smart contracts that are currently live. However it is not Turing Complete, meaning that a person cannot customize it as desired. Instead, the user must use the existing templates. Ethereum, introduced above, is a public blockchain platform which is currently the most advanced smart contract enabled blockchain. With a "Turing Complete" coding system, theoretically a user can put any logic into an Ethereum smart contract, and it will be run by the whole network. There are mechanisms in place to prevent abuse, and users need to pay for compute power by passing in "ETH" tokens, which act as payment for the miners who run the code.

[0152] There are some questions and challenges with respect to the structure described above. For example, decentralization is expensive. The more computers that run code, the more expensive things get for the end users. A system that has 10,000 computers running the user's code can incur significant costs: the computer operators are likely not going to provide their computer infrastructure and run the code on their computer infrastructure free of charge. In a public network, the users pay to run all the machines on the network. Having every computer ("node") in a system stores data (e.g. a copy of the blockchain, or a portion of the blockchain corresponding to specific assets/portfolios) and run the smart contract code embedded within the blockchain is more expensive than having one or two participants run the code on individual computers.

[0153] It is typically sufficient to have the code written on a blockchain so the parties can see the smart contract they are committing to. In this case, the code can be run privately, perhaps by the very parties to the transaction. This would save on compute costs, but increase risk because only the transaction parties would be verifying the transaction/contract action (whereas normal blockchain interactions are verified by anonymous servers).

[0154] Another factor in contracts is optionality. In many contracts, clauses are written into things to create a channel for arbitration. For example in a flat rental agreement, wear-and-tear from tenants is acceptable, but major damage needs to be repaired. How does code define these things? Force majeure is present in many contracts to allow for wiggle-room for the parties involved. In a smart contract environment, how does one party call that without abusing it or referring to a human arbitrator? These issues can be addressed through smart contracts. Ultimately, shared ledgers will have a role to play in removing the need for trust among multi-party agreements. Smart contracts make sense for all parties by reducing operational risk, and can provide automated trustworthy workflow between parties without a central specific coordinator. However, in the disclosed application, there is no need for any type of subjective arbitration since crypto-currencies pricing and trading data is highly public, replicated, and simple to understand. Any two

observers can easily agree on the pricing data coming from the API of a public crypto-asset exchange.

[0155] The following description provides details of an application from a user's perspective. FIG. 2 illustrates the overall process 200 for an example application in accordance with various aspects of the disclosure. By way of example, FIG. 2 shows a series of steps implemented by a portfolio index swap marketplace (PRISM) to create a portfolio of assets. Every portfolio index swap marketplace created can be a swap of one or more assets. In the marketplace, there are at least two parties to the trade and a smart contract 206. The parties are the portfolio buyer 202 and the portfolio seller 204. For clarity, throughout the rest of this document, the portfolio buyer may be referred to as the portfolio buyer, the buyer, the PB, or the user collectively; and the portfolio seller may be referred to as the portfolio seller, the seller, the PS, or a business entity.

[0156] Solidity, the Ethereum Virtual Machine and the Ethereum blockchain collectively work to enforce the smart contract that allows a single trustless portfolio to exist and execute. Swaps represent a type of promise, or agreement, for each party (the buyer and the seller) to swap asset exposures for some amount of time. With swaps, the underlying assets are not exchanged. All that is exchanged is a contract that binds each party to pay one another based on the change in value of the underlying assets as recorded by some agreed upon external data, like a benchmark, price feed, or in this case, the multi-validator oracle 210.

[0157] The multi-validator oracle 210 is an ensemble of blockchain-based smart-contracts and a set of applications (e.g., JavaScript based applications) that enable the validation and authentication of data sourced from the public APIs of any website in the world. This data is pushed to a blockchain-based smart-contract called the oracle (or any other name with similar functionality) 210. The oracle 210 is used to provide information to decentralized applications. More specifically, the oracle 210 can provide a data feed to the smart contract 206.

[0158] The trade can be explained in the context of a contract for difference (or CFD) example. Assume Alice wants to bet on the change in next quarter's US GDP. She creates the buyer portfolio 202 and the terms of the smart contract 206. For example, she creates a smart contract that includes a formula like $PAYOUT = ((ALICE_PREDICTION - GDP1Q2014 / GDP4Q2013 - 1) * GEARING)$ and funds it with 10,000 ETH. The script in the contract specifies that anyone who sends 10,000 ETH to this contract 206 will take the other side of this trade. Say Bob creates a seller portfolio 204 and accepts the offer. The script also contains the public key of an "oracle" 210, e.g. a trusted website that publishes economic statistics for the purpose of authoritatively fixing the settlement value of CFD's. After X days, the script in the contract 206 consults the oracle 210, pays a small fee to the oracle 210, and gets signed value for GDP1Q2014, which the script checks against the oracle's public key. Script then computes the formula and sends Alice $\max(10000 + PAYOUT, 0)$ ETH and Bob $\max(10000 - PAYOUT, 0)$ ETH.

[0159] The smart contract 206 swap mechanism enables two parties, a "buyer" and a "seller," to establish opposite positions in a trustless swap arrangement for exposure to a set of arbitrary assets. One of the problems addressed by the smart contract structure is that financial swap arrangements require trusting one or more parties to deliver or satisfy

obligations upon settlement (performance). Every swap thus has counter-party risk, and this risk must be priced and paid ultimately by one or more parties in the arrangement.

[0160] The smart contract herein provides a technical solution to this issue. The following is a brief summary of the process with reference to FIG. 2. The blockchain-based smart contract **206** is a trustless multi-asset swap mechanism (TMASM) paired with a data-feed oracle **210**, in which performance and settlement of a swap arrangement occurs automatically and without risk to either party. The swap is established by the buyer **202**, who selects one or several assets to which the buyer seeks exposure. The swap is accepted by the seller **204**, who agrees to hold the other side of the same position. Stated in another way, the buyer **202** offers a contract to the seller **204**, in which buyer **202** promises to pay the seller **204** the “fixed leg” of the swap, in return for the seller **204** paying the value of the portfolio (the “floating leg”) when, and only when, the contract is closed by the buyer **202**. The buyer is making a promise to the seller to pay them a certain amount which is the portfolio’s value at the time of the purchase. The time of the payment will be in the future. Assume, at the time of the purchase, the portfolio value is 100 ETH. The fixed amount is therefore 100 ETH that the buyer promises to pay the seller upon closing out the contract in the future. In swaps language, the buyer **202** has offered a contract to the seller **204** wherein the buyer **202** promises to pay the seller **204** 100 ETH (aka, the “Fixed Leg”), in return for the seller **204** paying the buyer **202** the value of the portfolio (aka, the “Floating Leg”), when—and only when—the contract is closed by the buyer **202**. The buyer **202** can choose to close the contract at any time, but the seller **204** cannot. Terms such as “close”, “settle”, or “liquidate” can mean that the buyer **202** has elected to close the contract and retrieve their portion of the settlement funds from the contract.

[0161] The seller **204** receives the buyer’s **202** request/offer and calculates the price of the contract and sends that information back to the buyer **202**. The calculation can be considered a service fee or commission charged by the seller **204** for entering the contract with the buyer **202**. The service fee (say 1 ETH) will need to be calculated for each portfolio which any user wants to purchase, since portfolios can be different in terms of their respective coins, the total investment amount, and other variables.

[0162] The buyer **202** accepts the terms of the swap contract by sending 125 ETH to the smart contract. 100 ETH pays for the investment in the portfolio, while the other 25 ETH can be the excess collateral required for the trade. In one aspect, if there is a service fee of 1 ETH, 24 ETH can be the collateral and 1 ETH can represent the service fee, for a total of 125 ETH. The buyer **202** effectively signs the contract using the private keys of the Ethereum address from which the 125 ETH were sent. The seller **204** is notified that the contract has been “accepted” by the buyer **202**. The seller **204** also accepts the contract by sending 125 ETH to the smart contract. The smart contract is now holding 250 ETH, and will continue to hold the 250 ETH until the buyer **202** chooses to close the contract, at which time the contract will be settled.

[0163] Upon acceptance by both parties, both the buyer **202** and seller **204** submit a blockchain transaction into the smart contract **206**, which includes the sum of a principal amount (derived from a value of the selected assets) and a

collateral amount (derived from the preferences of the buyer and/or seller). In one aspect, the transaction is the acceptance.

[0164] Upon receipt of both transactions, the swap contract **206** is activated, existing solely on the blockchain (for example, Ethereum’s blockchain). The smart contract **206** utilizes a data feed (oracle **210**) to periodically calculate the value of each party’s position. The buyer **202** may close the swap contract at any time, at which point the smart-contract will transact proportionate assets to the buyer **202** and the seller **204** (divided from among the principle and collateral deposited by both parties), based on the rise or fall of the underlying portfolio.

[0165] The portfolio buyer **202** has been able to enter that entire portfolio of assets without actually having to go out and purchase them and take physical custody of the assets. The process reduces the exposure of risk to the buyer **202**, and increases the convenience and speed of obtaining exposure to those assets. On the back end, the portfolio seller **204**, which can be an individual or a business entity, will act like a clearinghouse, but without taking custodial ownership of the assets. The portfolio seller can design and implement the smart contracts which anybody can see. The seller can take a fee for creating and managing the smart contract and participating in the selling process.

[0166] As noted in the example above, the portfolio buyer **202** contributed 100 ETH into the smart contract **206** as principal, plus 25 ETH as excess collateral. The seller **204** contributes 125 ETH at the same time. This is the seller’s acceptance of 100 ETH plus 25 ETH collateral. The contract **206** then holds 250 ETH **208**.

[0167] The seller **204** can hedge its position in the contract by purchasing the underlying coins that make up buyer **202**’s portfolio from an array of crypto-asset exchanges and then holds them in their own wallets. In this case, the seller **204** would go and purchase X litecoins, Y ripples, and Z dash, and hold them until the contract is closed by the buyer **202**, at which time the seller **204** can sell them on the open market. As a result, the seller **204**’s exposure to this contract is perfectly hedged at all times in the future.

[0168] Now, fast forward to the time when the buyer **202** decides to close the contract or ‘rebalance’ the portfolio. Assume that the time (T) is 7 days. For simplicity, the disclosure will assume that the buyer **202** wants to close the contract and not rebalance. When the buyer **202** chooses to close the contract, the closure is initiated via a signed message from the buyer **202**’s private key, which the smart contract **206** recognizes as the only signal that can initiate settlement of the trade. The smart contract **206** then settles the contract as follows. The smart contract **206** calculates the current value of the portfolio based on crypto-asset pricing data from the Oracle **210**. In this example, at T=7, let’s assume the value of the buyer’s portfolio is 150 ETH. That means the value of the portfolio went up 50% from its initial value of 100 ETH. The smart contract **206** calculates the amount of collateral and resulting proceeds to be sent to the buyer **202**, while subtracting the seller’s **204** commission for the trade (which, in this example, will be 1 ETH). The settlement calculation can be: at T=7 the buyer **202** will receive 174 ETH, while the seller **204** will receive 76 ETH. The buyer gets 174 ETH=100 ETH (original investment amount)+50 ETH (profit from portfolio appreciation)+25 ETH (original excess collateral amount)–1 ETH (the seller’s commission). The seller gets 76 ETH=50 ETH (difference

from original investment and loss on the trade)+25 ETH (original excess collateral amount)+1 ETH (the seller's commission) Once this calculation is finished, the smart contract 206 allows the buyer 202 and the seller 204 to withdraw their settlement amounts from the contract. This contract has now been settled.

[0169] Next, the seller 204 lifts the contract hedge by selling underlying coins or assets. Based on this example, the seller 204 lost 50 ETH as the result of its bet on the price movement of the buyer's 202 portfolio of assets. However, since the seller 204 purchased an exact mirror image of that portfolio of assets when the trade was initiated, when the trade is closed and settled, the seller 204 will sell that exact portfolio of assets for which it has earned a profit of 50 ETH. Therefore, the seller 204 has not lost any value on the trade, but has gained 1 ETH (the commission). This is why the seller 204 was perfectly hedged 212 at the start of the trade.

[0170] The process is "trustless" because it is executed automatically by smart-contract code on a blockchain. There is no entity or group of individuals that can alter the terms of the contract or manipulate its settlement. The smart contract always meets its obligations. The disclosed concepts herein bring the marketplace directly to the consumer in an entirely unique and novel way.

[0171] Normally, with a mortgage or a credit card, there are a number of different entities involved, including a bank. There are different processors, counterparties, etc. that can be holding the money at some point. By contrast, the approaches disclosed herein transform the financial system into a blockchain smart contract that is packaged in a consumer product that delivers to the consumer what the consumer wants—which is the ability to own a portfolio of assets and then also have the assets managed. In one aspect, a user can develop such a portfolio (for example, a portfolio of several blockchain assets or cryptocurrencies) without having to download an individual wallet for each blockchain asset currency to buy that currency or leave the asset on deposit with a counterparty. Consequently, the user can develop a portfolio of assets without requiring multiple private keys, multiple applications, and multiple counterparties. The user does not have to send money to each exchange to buy blockchain assets in that type of currency.

[0172] Thus, the concepts disclosed herein allow a user to create a multi-asset portfolio (e.g., a portfolio of multiple blockchain assets and/or any other type of asset). For example, the concepts disclosed herein can aggregate the process so the user can, in simple, fast, and efficient way obtain a portfolio of blockchain assets without downloading an individual wallet for each type of asset or leaving assets on deposit with a counterparty. In some examples, the management component can be accomplished through a leader-follower functionality, as further described herein. The end users can access such a service through a user interface, an application, a website, an API, or any other means.

[0173] FIG. 2 in connection with other figures, referenced throughout the disclosure, further demonstrates an example process. In the figure, reference to ETH is representative of any value/type of currency (including Bitcoin) depending on the platform desired. As noted above, in the first step, the user (buyer) creates their portfolio 202 of blockchain assets. The user can create the buyer portfolio 202 via an interface, such as web interface 300 shown in FIG. 3.

[0174] The web interface 300 can include a portfolio settings area 310 that allows a user to enter various parameters for building the buyer portfolio 202. The user can enter various parameters in the portfolio settings area 310 of the web interface 300 to create a portfolio. For example, the user can select blockchain asset components and respective asset percentages, such as A % Litecoin, B % Ripple, and C % DASH, from an assets area 304 and a percentages area 308. The user can provide an investment amount 302, such as 1 ETH, for the portfolio. The system can then calculate respective assets amounts based on the prevailing rates from the exchanges 214 coming from the blockchain asset data feed (the Oracle) 210 at that point in time, such as X number of litecoins, Y number of ripples, and Z number of dash, and provide the respective asset amounts in an asset amounts area 306. The user can then select a control element 312 to create or purchase the portfolio. For example, the user can click on the control element 312 to build the portfolio. In some cases, the control element 312 can include one or more labels, such as "create portfolio" or "purchase portfolio".

[0175] The system can summarize the portfolio with an asset value, a collateral value, a creation fee and a monthly fee as well. Any combination of this data can be presented. The user can then enter their ETH address to continue, which should be the same address used to send funds to the multi-asset portfolio. If the currency used for payment is ETH, the user could send the number of ETH to the ETH address. A QR code could also be used to summarize the transaction. A bar can be included in the graphical interface to instruct the user of their progress in how many ETH have been sent and how many are owed to complete the payment. Once payment is complete, a summary page can show the amount paid, an email address for a receipt, and a link to enable the user to view the portfolio they just created and purchased. In some example interfaces, the system can present a listing of portfolios created by a particular owner (ETH address). Data can include a starting value in ETH (or other currency) and dollars, a current value, a listing of the portfolios by name, date created, rank, a graphic like a pie chart for example, a Prism ID, a starting value, a current value, a rate of return, and/or a total value.

[0176] A further graphic can be presented which includes a left-to-right graphic of value in dollars or ETH that shows an initial value in one color and another increase in value in another color. In some examples, the interface can enable the owner of the portfolio to rebalance, sell or make the asset group public.

[0177] FIG. 4 illustrates an example interface 400 providing a purchase summary for the user and information for finalizing the purchase. The interface 400 includes a purchase summary 402 indicating various details of the purchase, such as a respective portfolio value, a respective asset, a respective asset percentage, and a respective asset amount. The interface 400 can also include a timer 404, which can indicate an amount of time left to pay. The interface 400 can also include a payment amount and location 406 identifying a payment amount and location (e.g., how much to send for payment and where to send the payment), in order to complete the purchase. Based on the interface 400, the user can determine the amount of payment necessary and address for sending the payment, which can be provided via the payment amount and location 406, as well as the amount of time left for transmitting the payment, as indicated in the timer 404

[0178] The interface 400 can also provide additional information for the user. For example, the interface 400 can include a notes and/or warnings area 408, which can present any notes, details, or warnings for the user which pertain to the purchase. In some cases, the interface 400 can also include a reference to other payments to complete the contract, such as 1 ETH for collateral or other fees or charges.

[0179] Once the portfolio is paid for and in an “active/open” state, the buyer can choose to close the contract at any time, but the seller cannot. The terms “close”, “settle”, “liquidate”, or “exit” each generally mean that the buyer has elected to close the contract and retrieve their funds from the contract. In one example, the buyer may commit to close within a specific period of time such that the seller is not held to the contract indefinitely. Notices can be provided to the buyer if they do not act within a predetermined period of time such as 6 months. In cases where a buyer passes away, procedures would be put in place to handle an estate or other entity closing the contract. In another aspect, the seller can have the right to initiate the close of the contract, purchase an option to close the contract or to suggest closing the contract which can be accepted by the buyer.

[0180] Referring back to FIG. 2, in step 2, the seller calculates and quotes a portfolio price to buyer. The seller calculates the cost of the asset group based on the buyer’s offer and sends that information back to the buyer. Additional details of this calculation will be discussed below. This calculation can be considered a “service fee” charged by the seller and can be calculated for each portfolio which any user wants to purchase.

[0181] In step 3, the buyer accepts seller’s quote and enters the contract. In this example, the buyer accepts the terms of the swap contract by sending 125 ETH (or other value or currency) to the smart contract 206. In one example, 1 ETH pays for the investment in the portfolio, while the other 24 ETH is the excess collateral required for the trade. These numbers are purely exemplary—however, most transactions can require one or more of the following portions: (1) the trade itself, (2) the cost of performing the trade, and/or (3) the excess collateral required for the trade. The buyer 202 “signs” the contract using the private keys of the address from which the 125 ETH were sent.

[0182] Referring to FIG. 5, an interface 500 can be presented to provide payment details and receipt information. The interface 500 can provide a portfolio number label 502 with the number 506 for the portfolio, as well as an email receipt input field 504 to allow the user to enter an email address for receiving a receipt of the purchase.

[0183] Turning back to FIG. 2, the seller is notified that the contract has been “accepted” by buyer 202. The seller 204 also “accepts” the contract by sending the seller’s corresponding amount to the smart contract 206. In this example, the seller 204 accepts the contract by sending 125 ETH to the smart contract 206. Again, the private keys of the sending address are seller’s signature keys for the contract. The address for an Ethereum contract can be deterministically computed from the address of its creator (sender) and how many transactions the creator has sent (nonce). The nonce can be an arbitrary number that is used once for a cryptographic communication. The sender and nonce are RLP (recursive length prefix) encoded and then hashed with Keccak-256. Below is an example pyethereum code:

```
[0184] def mk_contract_address(sender, nonce):
```

```
[0185] return sha3(rlp.encode([normalize_address
(sender), nonce]))[12:].
```

[0186] Below is a specific example with some discussion:

[0187] For sender 0x6ac7ea33f8831ea9dcc53393aaa88b25a785dbf0, the contract addresses that it will create are the following:

```
[0188] nonce0=“0xcd234a471b72ba2f1ccf0a70fcaba6
48a5eecd8d”
```

```
[0189] nonce1=“0x343c43a37d37dff08ae8c4a11544c7
18abb4fcf8”
```

```
[0190] nonce2=“0xf778b86fa74e846c4f0a1fbd1335fe8
1c00a0c91”
```

```
[0191] nonce3=“0xffff933a0bc612844eaf0c6fe3e5b8e
9b6c1d19c”
```

[0192] The smart contract 206 is at this stage holding 4 ETH, indicated by the amount 208 in FIG. 2. The smart contract 206 will continue to hold the amount 208 until the buyer chooses to close the contract, at which time the contract will be settled.

[0193] In optional step 4, the seller 204 can hedge the contract 212. This is an optional choice as part of the process and is not required to manage the portfolio. The seller can “hedge” its position in the contract by purchasing the underlying assets or currency that make up the seller’s portfolio from an array of blockchain asset exchanges 214, and then holding the purchased assets or currency in the seller’s own wallet(s). In this case, the seller 204 would purchase X litecoins, Y ripples, and Z dash, and hold them until the contract is closed by buyer 202, at which time they will sell them on the market. As a result, seller’s exposure to this contract is perfectly hedged (or nearly perfectly hedged) at all times in the future.

[0194] FIG. 6 illustrates a summary of the portfolio with an interface 600 providing a summary of the portfolio. The interface 600 includes a current value 602 and data such as the percentage of gain or loss. Feature 604 provides options for the buyer to sell, rebalance, share, or add to leaderboards. At some point, the buyer will decide to close the trade or rebalance the portfolio using the interface 600. Assume this is time (T)=7 days.

[0195] FIG. 7 illustrates an example method for creating a customized smart contract. In this example, the method includes receiving one or more parameters input by a user via a user interface for a customized smart contract (702). For example, the one or more parameters can include a first parameter and a second parameter associated with a creation of a customized smart contract. The method can include authenticating the one or more parameters via a public/private key associated with the user (704) and deploying the customized smart contract onto a blockchain (706).

[0196] The method can include generating the customized smart contract to implement the one or more parameters. Using an intermediary smart contract (which can be a contract creator), the process can be transparent and trustless, as the final smart contract (which varies per user) holding the funds or assets is, itself, created by a preceding contract that is the same for all users.

[0197] The customized smart contract can be deployed on a blockchain. The smart contract can run without a custodial risk, such as a risk of loss of securities held in custody occasioned by the insolvency, negligence or fraudulent action of the custodian or a sub-custodian. The reduction/

elimination of custodial risk can apply to other aspects of this application and not just the contract creator.

[0198] The smart contract is between a first party and a second party. In this example, no third party holds custody of the assets associated with the smart contract or the smart contract itself. Non-limiting examples of the one or more parameters include parameters associated with one or more auctions, wallets, real estate transactions, stock trades, alt-coin trades, crowdfunding, contracts, legal services, currencies, and so forth. The customized smart contract can be coded, stored and replicated and a distributed platform.

[0199] FIG. 8 illustrates an example method for providing a multi-validator oracle similar to oracle 210 shown in FIG. 2. The example method can include receiving a notification, at a multi-validator oracle, from an external smart contract, requesting data from the multi-validator oracle to be provided to the external smart contract according to a set of parameters to yield requested data (802). The notification or call from the external smart contract can be based on a trigger, such as a rebalancing notice, a message, market data, a user request, a market condition or threshold, an event, and so forth.

[0200] Based on the notification, the method includes providing the requested data to the external smart contract (804). Here, the multi-validator oracle can gather the requested data from at least one public application programming interface for a website that provides information associated with the requested data. In some examples, the multi-validator oracle can also gather at least a portion of the requested data from one or more other sources, such as databases, via respective calls for example.

[0201] In one implementation, two validators are used. However, in a generalized version of the multi-validator oracle, the system can add or subtract validators. An example of M of N validators can produce the same result for the oracle data to be considered “truth”. The method can also include multi-validating the requested data based on a first verification from a first private key and a second verification from a second private key (806).

[0202] The multi-validation can include requiring at least two validations from three or more possible validations. The multi-validation can include requiring a subset of validations from a superset of validations, where the superset of validations is larger than the subset of validations. In one example, each validator can pull the raw source data from the public APIs of each exchange, perform one or more calculations on it, and return the result(s). The data, and calculation result, is validated (which means it can be used for settlement) when both validators produce the same result and agree that the result is the “truth”. The reason why the system uses this process is that smart contracts themselves can’t make calls to public APIs and so they operate in a “walled garden”. Smart contracts can only receive information via messages from a public blockchain address or another smart contract. Thus, the owner of that address has to first get the data from the public APIs and then push that data to the oracle smart contract. The issue is trust in the person who pushes the data to the oracle as they would be able to push false results.

[0203] In one aspect, the smart contract runs on a public blockchain for the smart contract. The multi-validator oracle can also include or be a blockchain based smart contract. The requested data can relate to a benchmark, a price feed, an exchange rate, etc. In one aspect, the method includes

validating and authenticating the information received from the public application programming interface.

[0204] The buyer, in addition to being able to close out the contract, can also perform a rebalancing. An example interface 900 for implementing an example rebalancing option is presented in FIG. 9. The need for rebalancing arises in the context of a multi-asset portfolio in which the user may desire to change at least one asset’s percentage or amount in the portfolio after it is created. When a user wants to change the composition of their portfolio, like adding or removing a coin, or changing the weightings/percentages of a coin in their portfolio, the user is able to use the same asset group to do so.

[0205] Interface 900 of FIG. 9 shows a current value 910, the current assets 912, an option to change percentages 906, 908, and an option 902 to complete the rebalancing. An option 904 for accessing a rebalancing history can also be provided. When a rebalance occurs, a similar set of calculations takes place as when the user created the portfolio or wants to settle the portfolio (e.g., a calculation of respective assets amounts based on the prevailing rates from the exchanges coming from the blockchain asset data feed at that point in time, etc.).

[0206] A trustless multi-asset swap rebalancer is a set of functions programmed into the blockchain-based smart contract 206 which allows for the rebalancing of a portfolio of assets in accordance with the requirements and demands of the owner of that portfolio. These assets exist via the trustless multi-asset swap mechanism technology and the trustless multi-asset swap rebalancer is a component of that technology.

[0207] An example trustless multi-asset swap rebalancer process using interface 900 can be as follows.

[0208] The buyer 202 decides which coins it would like to change in the interface 900, and sends the new portfolio composition to the smart contract as a signed message. The smart contract 206 calculates the new portfolio value, collateral, etc. based on oracle data 210 at that point in time. The seller 204 takes a rebalancing fee to process this change and accepts the new portfolio. The seller and buyer do not have to send more collateral if the rebalance is within the collateral limits for the contract. If the user 202 rebalances to a large portfolio, the user would need to add collateral to the contract. This is one of the reasons why in the aforementioned example, the system asks the user to send “excess collateral” to the smart contract at the creation of the contract. This excess collateral can be used for future rebalances as well as other purposes.

[0209] Both parties can confirm the rebalance data via a signed message, after which the state of the portfolio is updated. The seller can change its underlying hedging portfolio accordingly to maintain the seller’s perfect hedge by buying or selling the mirror image of what the user bought and sold during the rebalancing of the contract.

[0210] The contract is written into the blockchain and thus onto the public ledger which may not be manipulated by any single party. When the user decides to rebalance the contract, or the percentage of one or more assets in the portfolio, that results in a state change. The new state change is going to also be written into the blockchain. The new state change cannot be erased or changed by any participant, and every participant can see it.

[0211] When a user rebalances the portfolio, the rebalancing can result in various scenarios. For example, if a user

rebalances the portfolio to have a higher value than its current portfolio value, then the user would have to add incremental funds to the smart contract. In one example, assume a portfolio that started out as a 100 ETH valued portfolio is rebalanced and in the rebalancing becomes worth 200 ETH. The buyer must put in an additional 100 ETH. The seller also must put in an additional 100 ETH. The amount added at the time of the rebalancing is that value above the current value of the portfolio (not the original value).

[0212] The portfolio is fully collateralized. One of the risks in derivatives is that every participant is buying them on margin. If that margin gets exhausted, then problems arise. By contrast, the contracts disclosed herein are fully collateralized, which reduces the entire counterparty risk or trust in the system.

[0213] If the seller does not want to add more collateral, then the rebalancing would not be established. If the user wants to change the allocation of cryptocurrency and the value of the portfolio does not change, the rebalancing can occur without additional collateral or approval from the seller. However, there could be mechanisms where if thresholds are met, then the seller would have to approve the rebalancing. For example, if more than 20% change in any asset is made, then the seller can be required to approve. In another scenario, if the seller does not want to add collateral, the buyer could go back to the marketplace and a different seller could buy into the rebalanced portfolio and the portfolio could end up with two sellers, each having a proportional share of the portfolio. However, if the seller is a business entity that is servicing the portfolio sale requests, then those rebalancing requests can always be met (within limits). Thus, such a business entity could create the entire marketplace for such trades. The sellers can be more like hedge funds and larger players or can provide a means for such players to get access to the smart contracts disclosed herein. After the user rebalances the portfolio, the process goes back to normal until the user closes out the contract. The user can rebalance any number of times before closing. At closing, as instructed by the buyer, the smart contract calculates the result and sends the right amount to each party.

[0214] FIG. 10 illustrates an example method for rebalancing a multi-asset portfolio. The method includes establishing a multi-asset blockchain-based trustless smart contract for managing a multi-asset portfolio (1002), receiving an indication from an individual associated with the multi-asset portfolio of a desire to rebalance the multi-asset portfolio (1004), presenting the individual with an interface to rebalance the multi-asset portfolio (1006), receiving, via the interface or some other means, input from the individual requesting a rebalancing of the multi-asset portfolio (1008), and updating the multi-asset blockchain-based trustless smart contract to yield a modified multi-asset blockchain-based trustless smart contract which incorporates the rebalancing (1010).

[0215] Technically, the smart contract could be the same contract, existing in the same place on the blockchain, but with a new “state”. That new state reflects the new composition of the portfolio which is recorded in the blockchain and is based on the rebalancing. The blockchain is the transparent and relatively immutable record of state changes which the contract undergoes. The system writes a new entry in the blockchain that reflects the new portfolio composition.

In another aspect, the system could create a new contract but with different parameters (asset allocations). The old contract can be discarded, deleted or made inactive.

[0216] The indication of the desire to rebalance the multi-asset portfolio can include a request to rebalance one or more assets in the multi-asset portfolio or add/delete one or more assets in the multi-asset portfolio. The rebalancing of assets can involve a modification in a percentage or an actual number of coins (or real estate, or other asset). The interface allows the user to enter a number of coins or percentages with one or the other updated instantly based on the prevailing exchange rates and the current value of the portfolio. A user cannot rebalance to a greater amount of value than the current contract contains unless the user posts additional collateral. Depending on what the rebalancing structure is, the method can include receiving additional collateral associated with the multi-asset blockchain-based trustless smart contract to create the modified (or new) multi-asset blockchain-based trustless smart contract.

[0217] Typically, the individual is the buyer of the multi-asset portfolio. However, the individual could also be, in some scenarios, the seller, the management entity, some other third-party, or a combination of individuals or entities that, through some mechanism, agreed to a rebalancing. The method can include receiving an authorization from a seller of the multi-asset portfolio for the rebalancing prior to creating the modified/new multi-asset blockchain based trustless smart contract.

[0218] The method can further include calculating at the multi-asset blockchain-based trustless smart contract an updated value to apply to the new multi-asset blockchain-based trustless smart contract based on data received from an oracle which provides real-time valuation data. The authorization received from the seller can include a signed message which results in a change in state of the multi-asset blockchain-based trustless smart contract to yield the modified multi-asset blockchain-based trustless smart contract. As noted above, the modified multi-asset blockchain-based trustless smart contract can be the same contract but with the new data or parameters.

[0219] Referring to FIG. 11, an interface 1100 can enable social trading. The interface 1100 allows users to share their “portfolio” with their connections/friends in a social network. As illustrated, users can post messages 1102 via interface 1100 which include links, addresses, and/or other information for sharing assets and/or portfolios between users in the network. The user interface 1100 can be graphical, multimodal, speech-based, text based, graffiti based, or any combination of input modalities to achieve any function disclosed herein.

[0220] The social or social media aspect of this disclosure can also relate to the leader-follower feature. The leader-follower mechanism is to reduce and eliminate the risk associated with any individual giving up custody of one’s wealth to another person or corporation who acts as a trusted third party and whose goal is to make investments on behalf of the individual. The system can insert or include certain authorizations into a smart contract. Prior to the present disclosure, if people wanted someone to manage their investments, they go send their money to a financial advisor or entity. Such entities have infrastructure to keep their investments safe. The follower-leader function allows one portfolio buyer, who essentially has a contract, which is their portfolio, to authorize another contract to do a specific

thing—to send messages to their portfolio that tell it how to rebalance and tell it the exact percentages and the weightings. The leader contract, however, cannot instruct the follower contract to settle, or sell collateral, etc. It is only allowed to send a message to the follower of portfolio saying, for example: “I just rebalanced my portfolio and here are my new weightings and assets.” And then the follower portfolio automatically rebalances based on the leader adjustments. The leader portfolio never has to take custody of funds. This approach removes all the custodial risk aspects. The messages can be delivered to one or more destinations, including other smart contracts, or any social media outlet.

[0221] Signed messages relate to data that is transmitted from one key to another. The message is the data and the data can be anything (an instruction to rebalance, for example, or an instruction to settle the portfolio). It is a signed message because the data is signed with the private key of the public key from which it is sent. In this case, the follower portfolio has authorized the leader portfolio to send it messages (e.g., the data of the new portfolio weightings and coins) and as long as the message is signed with the leader’s private key, the follower portfolio will change its own allocations to that of the leader. Thus, as a security feature, smart contracts which interact with each other or with their users can do so through signed messages to provided authenticated data transmission.

[0222] The example above discusses implementation of asymmetrical cryptography (i.e., Public key cryptography) for secure or authenticated data transmissions. This example cryptography implementation is provided as a non-limiting example for the purpose of clarity and explanation. It should be noted that other encryption/cryptography configurations and techniques, such as symmetric-key algorithms, can also be implemented and are contemplated herein.

[0223] The reference in FIG. 11 is to Twitter. However, the principles disclosed can apply to any social media environment. For example, Facebook, Pinterest, Instagram, Snap-Chat, blogs, and so forth, are social networks through which individuals can share their portfolios. The individual functionality of these and any other social media are incorporated herein and applicable to the general concept of receiving a posting of a social media object from a user which includes a button, a hyperlink, or other type of interactive feature which points recipients of the posting to a portfolio of the user. Steps which must be implemented in order to enable recipients of the posting to interact with the posting and thus initiate a following within a recipient portfolio of the posting entities leader portfolio can occur in various ways.

[0224] Posting to social media can allow people to “find” a smart contract, and authorize that smart contract to serve it rebalancing instructions. The interface 1100 for such a process can be implemented in a variety of platforms and computer programming languages, such as web-based (e.g., HTTP or HTTPS), mobile or native applications, etc. Following a portfolio allows the user to create their own marketplace (PRISM) but also allow the leader to do portfolio allocation decisions for the user. In some cases, this can involve two aspects or processes, which can run simultaneously. One is the front-end process which displays the contract data for the user, which can reside on the blockchain in what is called bytecode, and the second is the actual smart contract itself, which can rely on other technologies. In some

examples, the front-end can be “read-only” for a smart contract data runs on the blockchain.

[0225] For example, a social media site may communicate via APIs or other means with portfolio management entities in such a way as the posting entity and the recipients can remain within the social media site to confirm a transaction associated with one portfolio following another portfolio. Transaction confirmations happen via the blockchain. The social platform can generate or hold keys for users to initiate a transaction securely. In some cases, a social media site can be modified to function as a client-side encrypted wallet (e.g., ETH wallet) and support such functionality.

[0226] In some aspects, a posting to a social media site can include a hyperlink which opens a new webpage when activated by the user, such as the marketplace webpage. Users of the marketplace can initiate transactions using a wallet (e.g., ETH wallet), such as Mist, MyEtherWallet.org, or any other wallet. In some configurations, when a user uses the marketplace, the client device of the user may have a separate window or tab open which can be the user’s wallet. The wallet can store the private keys the signed message(s) can thus originate from that wallet. In this regard, another aspect of this disclosure can be a blending of an interface with a wallet interface which stores, presents, and/or provides access to the necessary private keys within a social media network site or environment. Thus, from a user’s perspective, the user appears to remain within the social media site, but will also have access to the services available through the user’s wallet.

[0227] Third-party entities can also aid in the integration between the social media site and the portfolios of the posting entity and the following entity. This disclosure covers processing from the standpoint of any of these entities including the portfolios themselves. Thus, the transmitting, data, and/or coordination of data or information can be claimed herein from the standpoint of a social networking site, a third party, a portfolio, etc. Individual entities can also provide particular steps to affect at least a part of the process as well.

[0228] In some respects, the leader/follower functionality can function like programmed money. It is money that cannot be told to do bad things. In the hedge fund world, there can be solvency issues and audits, and there is no way around these issues as of now because of how investments are handled: Hedge funds need to take the custodianship of the funds. With the leader/follower approach disclosed herein, the user (1) Never gives ownership of the funds to the seller, (2) Never has to worry about the settlement calculation, and (3) Can get the information directly from a professional, the leader in this case. The program in the smart contract only is allowed to give the instructions of rebalancing.

[0229] The follower in some cases could be given options to accept the leader’s rebalancing, or to further adjust the rebalancing and then apply it to the follower’s own portfolio. Market research could be provided at this point to the follower. For example, the 1 year or 1 month percentages of the performance of the assets in the rebalancing could be presented to the follower before accepting the rebalancing. Followers could also program or require certain parameters or triggers that are automatically implemented. These can relate to performance, performance history, boundaries on variation in the rebalancing, thresholds on additional collateral needed, limits on percentages of certain assets in their

portfolio, a percentage of how much of the rebalancing to accept (e.g., I will follow 50% of the changes of the leader) and/or the like. The parameters may involve timing, such as implementing the rebalancing 1 week after the leader rebalances and with the authorization of the follower. In this way, the follower could see the performance of the rebalancing before following.

[0230] The use of blockchain and smart contracts herein can provide significant advantages over a robot-type advisor. For example, with a robot advisor, the user is still giving their money to a third party to manage. Use of a third-party custodian of the money can create various risks as previously explained. However, the smart contracts and rebalancing performed on the blockchain herein eliminates the risks of using a third-party custodian of the money.

[0231] The use of blockchain and smart contracts herein also provide significant advantages over use of brokers, and reduces risks associated with using brokers. For example, assume a broker has an algorithmic trading system that is a computer program that buys or sells assets. With the broker, the algorithmic trading system, and thus the computer program, is hosted and/or owned by the broker. The computer program instructs the broker's system to make trades on specific market(s) based on funds that the broker has taken custody of. By contrast, in the approaches herein, the blockchain is a computer program that is not hosted on systems owned by the broker or any single entity. Moreover, the smart contract algorithm is provided by computer code in the blockchain (e.g., Ethereum blockchain or any other blockchain). The blockchain is not owned by any single entity. The blockchain does not take custody of the funds, as the funds "exist" on the blockchain. Thus, it separates the custodial aspects from the investment manager. The follower still has to trust the leader's investment decisions, but the follower does not have to trust the leader's custodianship of the money.

[0232] FIG. 12 illustrates an example interface 1200 for accessing portfolio information and features. In this example, the interface 1200 can provide a summary section 1206 with a summary of the portfolio for the user. The interface 1200 can include a clone option 1202 to allow cloning of the portfolio. For example, the clone option 1202 can trigger a cloning operation, which can provide an opportunity for a user, such as a friend or acquaintance, to clone the portfolio.

[0233] The interface 1200 can also include a follow option 1204. The follow option 1204 can be selectable to allow a user to automatically rebalance when the owner rebalances the portfolio. The follow option 1204 can allow a user to follow changes made to another portfolio of choice. Thus, users can copy, clone, and/or follow one or more portfolios via interface 1200. A trustless leader-follower application enables a mechanism which automatically "copies" or "mimics" the actions of one trustless portfolio onto another, without the former taking custody of the latter.

[0234] One of the challenges addressed by the leader-follower application involves the issue that many investors either lack the ability or time to determine what assets are good investments. Such assets could include stocks, bonds, digital currencies, blockchain assets, commodities, and many other types of assets. Therefore, many individual investors outsource the management of their investment decisions (and the execution of investments) to the professional asset management industry. This can include financial

advisors, hedge funds, mutual fund managers, and many other types of investment managers.

[0235] Here is a simple example of where that will be helpful with basically non-digital assets, stocks or bonds. A lot of the regulation that exists on a national basis does not allow people to buy foreign stocks unless they are listed on a certain exchange using an ADR, or American Depository Receipt. This is essentially a reflection or a listing of that stock in the American markets. If a person wants to buy small or mid-cap stocks or some other esoteric asset in another country or jurisdiction, the amount of inefficiency and cost that the person is going to face in doing so is large. There are also limits on the amount that the person can buy. However, with the blockchain-based smart contract approaches and concepts disclosed herein, the technology could enable an easy purchase of such foreign assets. The disclosed concepts can enable micro payments with great fluidity of value transfer on a global basis, as well as significant freedom, lower costs, and higher security.

[0236] When individuals deposit their money with financial firms, they are exposed to a well-known set of risks. One of those risks is custodial risk, defined by the fact that the individual is no longer in possession of their own funds and has elected another legal entity—the firm—to be the custodian of those funds. This decision creates a risk for the customer because the custodian could intentionally or unintentionally lose/steal those funds. Historically, some of the ways custodians have lost customer funds include but are not limited to: (1) fraud; (2) negligence of the custodians; (3) improper or erroneous accounting (either intentional or unintentional); and/or (4) domiciling of those funds in high risk country which seizes those funds for political reasons. This is why a firm's reputation, track record, physical location, type of fund, and other factors play an important role in the asset management and investment industry.

[0237] Individuals must choose whom to "trust" as the custodian of their funds. Often, it can take decades for a firm to build trust and develop a "good reputation". However, this trust comes with a cost. The cost is an entire array of infrastructural processes designed to prove to others (clients, regulators, employees, etc.) that they are acting in good faith as custodians of their clients' money. Some of those infrastructural items include but are not limited to: (1) regular accounting and audits by reputable firms, (2) a corporate governance and legal structure designed to both adhere to local regulations and properly incentive the management, and/or (3) building up a track-record within the industry of competence, integrity, and good will. Reducing and even eliminating the cost of this trust is achieved with the leader-follower aspect of this disclosure.

[0238] When one user follows another user's portfolio, that user authorizes the user's portfolio to receive "rebalancing" instructions from the other user's portfolio; that is, one user is the follower and the other user is the leader. The leader portfolio is only authorized to send rebalancing instructions to the follower portfolio and not, for example, ask the contract to settle, remove the collateral from the contract, change the settlement address, or a whole number of other actions that are reserved just for the follower. There may be any number of parameters or functions that can be programmed into the contract that the followers can follow. Functions can include, for example, when to close the contract based on the leader functions performed. The leader-follower functionality can be a feature programmed

into the smart contract **206**. The leader-follower mechanism is a set of functions programmed into a blockchain-based smart-contract that allow the follower portfolio to receive instructions from the leader portfolio. One example of instructions can include cryptographically secure messages sent via a blockchain network from the leader's portfolio to the follower's portfolio. These messages contain the information for the follower's portfolio to rebalance to match that of the leader's portfolio, without the leader ever taking custody or being able to seize the funds of the follower. In other aspect, the leader could be enabled with the ability to not just send messages but also change follower portfolios. For example, once the leader has made a change, the market conditions might not be exactly the same when the follower wants to make the similar adjustment. The leader could cause the follower to make a less risky change than performing the exact mirror adjustment as the leader did. The leader's change might adjust the marketplace immediately which might put the followers in a less advantageous position than the leader.

[0239] Blockchain asset portfolio information from various users can be used to create leadership tables and rankings. For example, the system can use portfolio performance information to post leaderboards on a daily, weekly, monthly and/or yearly basis. FIG. 13 illustrates an example interface **1300** of a leaderboard. Leaderboards can allow users to "follow" specific "portfolio managers" with just a few clicks as described above.

[0240] Interface **1300** can display portfolio owners **1302** and portfolio data **1304, 1306, 1308, 1310, 1312** for corresponding portfolios. The portfolio data **1304, 1306, 1308, 1310, 1312** can include, for example, one or more portfolios or portfolio descriptions, a portfolio return value, a portfolio weekly return value, a portfolio monthly return value, a portfolio yearly return value, etc. The portfolio data **1304, 1306, 1308, 1310, 1312** can pertain to a single portfolio associated with a respective owner or a group of portfolios associated with the respective owner. Moreover, a portfolio associated with the portfolio data **1304, 1306, 1308, 1310, 1312** can include multiple assets, which can be homogeneous assets (i.e., same type of assets) or heterogeneous assets (i.e., different types of assets).

[0241] In some cases, the portfolio data **1304, 1306, 1308, 1310, 1312** can include more or less information, such as asset information, portfolio parameters, etc. The portfolio owners **1302** and/or portfolio data **1304, 1306, 1308, 1310, 1312** can be user-selectable. Thus, users can select a particular field or value to interact with that field or value. For example, in some cases, users can select a user from the portfolio owners **1302** to clone or follow the portfolio associated with that user. Thus, a user can review the portfolio owners **1302** and portfolio data **1304, 1306, 1308, 1310, 1312** and select portfolios to follow or clone based on the data provided in the interface **1300**.

[0242] The portfolio data **1304, 1306, 1308, 1310, 1312** can also be selected by a user to drill down on the associated data. For example, portfolio data **1304** can display a portfolio or portfolio description of a respective owner. The user can select the portfolio data **1304** to view additional details about that particular portfolio, such as portfolio parameters, user preferences, portfolio statistics, portfolio assets, following information (e.g., how many users are following that particular portfolio, which users are following that particular portfolio, comments provided by users regarding that par-

ticular portfolio, etc.), notes from one or more users pertaining to that particular portfolio, portfolio historical data, etc.

[0243] The portfolios presented in the interface **1300** can be ranked or organized based on one or more factors, such as performance statistics, ratings, number of followers, consistency, type of assets, averages, or values associated with the portfolio data **1304, 1306, 1308, 1310, 1312**. For example, the portfolios can be listed in order of performance based on one or more performance factors. As another example, the portfolios can be listed according to user rankings. In some cases, users can filter which portfolio owners **1302**, portfolios, and/or portfolio data **1304, 1306, 1308, 1310, 1312** to view on interface **1300**. For example, a user can filter interface **1300** to only display friends of the user (e.g., other users having a relationship or contact with the user), a selected list of users, a top number of users, users associated with a particular organization, users in a particular geographic location, etc. The user can also filter the interface **1300** based on other parameters or factors. For example, the user can filter the interface **1300** to only display portfolios having certain assets, portfolios that are older than a specific period of time.

[0244] The leader/follower feature enabled by interface **1300** can allow users to apply different investment research processes to portfolio design and construction. For example, a user could create the following exchange traded fund (ETF)-like blockchain asset product: A market cap weighted portfolio of the top ten blockchain assets by market cap. Such a portfolio is like the Dow Jones of blockchain asset investments. It would automatically rebalance over some period for which a user could choose. The user could also create a version with the top twenty coins or the top 50 coins. Users could back-test and publish results of different portfolio compositions. This could also work for an inverse market cap ETF or for making a portfolio allocation that optimizes return vs volatility using modern portfolio management techniques. A leaderboard can present one or more of the following data: a pie chart (or other graphic) illustrating a distribution of assets, an owner, a portfolio name, a percentage of returns on a lifetime basis, weekly basis, monthly basis, yearly basis, or user-selectable basis. A user interaction with one of the portfolios (such as on a pie chart graphic) can return a detailed accounting of the assets and the percentage of the portfolio that has that respective asset.

[0245] In this way, users could create investment strategies and publish research which supports their portfolio composition, and share their portfolios for other users to follow. The leader/follower feature also enables other applications. For example, a user can employ a specific pattern to significantly reduce the chance of a hack on the private key of the user's portfolio while maintaining the flexibility to trade that portfolio from an "insecure" computer or network. The user can create a "hot" leader portfolio which has a negligible amount of value in it, for which the private key can be exposed carelessly. That "hot" portfolio would be the leader to the same user's "cold" portfolio which holds the main value of the portfolio, say \$30 k or 40 k worth of digital assets. The private key to this "cold" portfolio would be generated in a secure environment and put in cold storage for the life of the portfolio. Since the hot/leader portfolio is only authorized to rebalance the cold/follower portfolio then if the "hot" key was compromised, the worst that an attacker

could do is simply rebalance the user's hot portfolio but never steal the funds in the user's cold portfolio.

[0246] In other examples, active trading of portfolios can be provided by those skilled in such endeavors on behalf of those with limited skills. For example, assume a user has a friend who has had a lot of success trading and asset management. However, the user is wary to let the friend trade a large number of coins due to legal, custodial, and/or accounting risks. With this approach herein, the user can just follow that the friend's portfolio without any of those risks, as the user retains control and does not relinquish custody of assets to the friend or any other third party.

[0247] In some cases, non-crypto assets can be provisioned by adding new oracle data feeds. For example, by connecting this technology to a Bloomberg terminal and corresponding execution infrastructure at a broker-dealer, hedge fund, investment bank, etc., different types of non-crypto assets can be provisioned as or for portfolios. To illustrate, users can add gold or other precious metals to their trustless portfolios. In such an example, those real-world assets could act as a trustless entry and exit mechanism for selling off entire exposures of blockchain assets should the user choose to do so. For example, the user liquidates an entire portfolio and moves to gold directly within their trustless portfolio on the blockchain. All of this is profound in that the system can have "fully reserved" digital assets collateralized with blockchain assets such as Ethereum.

[0248] The global derivatives industry is vast with over a quadrillion dollars in outstanding contracts and hundreds of different types of derivatives like swaps, options, CDS, exotics and many more. In the traditional financial system, almost all consumer and business financing activities are in some way attached to a financial derivative, whether that be a credit card or mortgage (CDS and interest rate swaps), activities in the CFO's office of a fortune 500 (FX currency swaps and futures), or the trading operations at a large exchange (risk management and hedging). This branch of financial engineering can be combined with the disclosed blockchain technologies and smart contracts to produce consumer financial products that cost less, eliminate intermediaries, increase security, and provide entirely new services, such as the trustless portfolio service described herein.

[0249] The smart contract disclosed herein can apply to any contracts (e.g., legal contracts, insurance contracts, etc.). For example, the smart contract described herein can be applied to insurance contracts, business contracts, intellectual property licenses, specific performance contracts, assignments, buying or selling a product or services, and so forth. Indeed, the procedures, components and systems disclosed herein could apply to any kind of contract. Any kind of blockchain technology can also be implemented or combined with the concepts disclosed herein. In fact, the technologies disclosed herein are blockchain agnostic and can be implemented with any future blockchain technologies.

[0250] FIG. 14 illustrates an example method for leader/follower approach. A method includes establishing a multi-asset blockchain-based trustless smart contract for managing a multi-asset portfolio (1402), inserting into the multi-asset blockchain-based trustless smart contract an authorization for a leader multi-asset blockchain-based trustless smart contract to send messages from the leader multi-asset blockchain-based trustless smart contract to a follower multi-asset blockchain-based trustless smart contract regarding a leader

rebalancing of the follower multi-asset blockchain-based trustless smart contract (1404), and modifying the follower multi-asset blockchain-based trustless smart contract based at least in part on the leader rebalancing of the leader multi-asset blockchain-based trustless smart contract (1406). The modifying of the multi-asset blockchain-based trustless smart contract can include an automatic follower rebalancing of the multi-asset blockchain-based trustless smart contract that matches the leader rebalancing.

[0251] The modifying of the leader multi-asset blockchain-based trustless smart contract can include a follower rebalancing of the multi-asset blockchain-based trustless smart contract that follows a portion or a subset of the leader rebalancing. For example, the system could include various rules, policies, constraints, and so forth, which a follower could implement such that upon a leader rebalancing a leader portfolio, a policy could be implemented which compares the rebalancing to the policy. In some cases, the policy may require that there is no change to one particular asset in the follower portfolio. If the leader portfolio changed the percentage of that particular asset in the leader portfolio, then the follower portfolio can evaluate the changes, apply the policy, and implement a rebalancing that is not exactly the same as the leader portfolio but implements aspects or principles set forth in the leader portfolio. In other cases, however, the follower may have the option to apply the rebalancing with the change to the particular asset. For example, the follower can approve such modification or include one or more policies or conditions for approving such modification.

[0252] To illustrate, the leader portfolio may have doubled a percentage of a second asset in the portfolio, as well as doubled a first asset. The policy may require no change in the allocation of the first asset. Thus, the follower portfolio may increase the balance of the second asset by 150%, rather than the doubling as occurred in the leader portfolio. The policy may require that the rebalanced follower portfolio have the same overall market value as the leader rebalanced portfolio and thus implement similar changes to those of the leader, although in different proportions, to arrive at the same overall market value. There are other scenarios which can be contemplated in how to structure a policy which would be evaluated against a leader portfolio change. Parameters which can be implemented in such a policy can include an overall portfolio value, percentages above or below and overall rebalanced portfolio value, parameters associated with individual assets, parameters associated with timing, such as when to implement a rebalancing, or how to implement portions of a rebalancing over time, parameters associated with external data which can be accessed and evaluated to confirm whether to follow a leader portfolio or whether to decline to follow, personal profile data, social networking data from individuals and/or social networking connections, a level of a social networking connection, and so forth. The embodiments in this regard can be claimed from a leader perspective of transmitting the leader changes to one or more followers, or could be claimed from the follower perspective in terms of receiving the leader information, and making adjustments accordingly.

[0253] The leader multi-asset blockchain-based trustless smart contract can send rebalancing messages to the follower multi-asset blockchain-based trustless smart contract. The method can include receiving market data at the multi-asset blockchain-based trustless smart contract prior to

modifying the multi-asset blockchain-based trustless smart contract such that the modifying takes into account both the market data and the leader rebalancing. In one aspect, the multi-asset blockchain-based trustless smart contract is modified for a percentage of the leader rebalancing. Further, there can be timing variations to the process. The method can include modifying immediately or in a predetermined time after receiving the leader rebalancing. There can also be events and conditions implemented in the process. For example, the method can include modifying the smart contract upon occurrence of a specific event, such as a user input or a triggering condition. In another example, if the follower receives the data late, such that the market has adjusted to a large leader modification, then the follower can take into account the current market conditions, the timing, and so forth to implement the same change or a variation thereof.

[0254] FIG. 15 illustrates an example interface 1500 for selling a portfolio of assets. The interface 1500 can include a summary view 1502 which can display assets 1510 and corresponding asset data 1512. Example assets 1510 can include, without limitation, Litecoin, Ripple, Ethereum, Dash, Bitcoin, and/or any other crypto and non-crypto assets. The asset data 1512 can include an initial value (e.g., value at time of purchase), a current value, a percent of the total portfolio investment or value, a current rate, a performance (e.g., gain, loss, etc.), and so forth.

[0255] The interface 1510 can also include a liquidation details area 1506 which can indicate one or more assets from the assets 1510 being sold, purchased, traded, etc., an amount of the one or more assets being sold, purchased, traded, etc., an address or party involved in the liquidation transaction (e.g., buyer, seller, etc.), as well as other information associated with the transaction, such as comments or preferences. The interface 1510 can an address 1504 associated with the liquidation details. For example, the address 1504 can identify the address where the smart contract should send the assets.

[0256] The interface 1510 can include a control element 1508 configured to trigger or execute the liquidation when activated by a user. For example, the control element 1508 can be a button that the user can select to liquidate the portfolio presented or selected, initiate the liquidation operation, accept the liquidation operation, and/or otherwise complete the liquidation operation. The smart contract, upon the buyer liquidating the portfolio, calculates the various amounts that are to go to the respective buyer and seller and carries out the liquidation operation based on the calculated amounts and the smart contract instructions.

[0257] The interface 1510 can allow a user to liquidate an entire portfolio or a selected subset of assets in the portfolio. In some cases, the interface 1510 can also allow the user to liquidate a group of portfolios. The user's specific preferences vis-à-vis which assets and/or portfolio(s) to liquidate can be specified via the liquidation details area 1506, for example.

[0258] FIG. 16A illustrates an example method for asset management in accordance with various aspects of the present technologies. The example method in FIG. 16A focuses on the operations of the smart contract, as opposed to the separate operations of the oracle, rebalancing component, leader/follower component, and so forth.

[0259] In this example, the method includes receiving, from a buyer and at a smart contract, an identification of a portfolio of assets (1602). Blockchain assets are one

example of such assets. Insurance contract, stocks, bonds, commodities, real estate, intellectual property, legal contracts, etc., are examples of other types of assets. The portfolio can include various types of assets, including various types of blockchain and/or non-blockchain assets. Moreover, the smart contract can be implemented in a blockchain, as previously described. The identification of the portfolio of assets can include a respective percentage of each type of asset to enter into the portfolio of assets.

[0260] The method includes receiving, from the buyer, an amount that the buyer will invest in the portfolio of assets (1604). The amount can be any amount specified by the buyer for investing in the portfolio. The amount the buyer will invest can include the current value of the assets. In some cases, the amount can be in cryptocurrency, such as Bitcoin or Ethereum.

[0261] The method includes receiving, based on one or more exchange rates (e.g., prevailing rates) and the amount being invested by the buyer, a number of assets in the portfolio of assets (1606). The number of assets can be the number of assets in the portfolio as calculated based on the amount the buyer is investing and the one or more exchange rates.

[0262] The method includes receiving a confirmation from the buyer of the portfolio of assets having the number of assets (1608). The confirmation can indicate that the buyer agrees with or validates the portfolio with the number of assets. The method includes and calculating, by an entity, a cost of the portfolio of assets to yield a contract (1610). The entity can be, for example a seller of the portfolio of assets.

[0263] The method further includes receiving the amount and/or excess collateral from the buyer as a buyer acceptance of the contract (1612). For example, the method can include receiving the amount invested by the buyer and excess collateral as a buyer acceptance of the contract. Moreover, the method includes receiving an entity acceptance of the contract by receiving an entity amount at the smart contract (1614). The entity can sign the entity acceptance using one or more entity signature keys for the contract, which can include, for example, private keys of a sending address associated with the entity.

[0264] In some cases, after receiving the entity acceptance, the method can include the entity purchasing the underlying assets that make up the portfolio of assets from one or more exchanges and holding the underlying assets.

[0265] The method includes receiving, at the smart contract, an indication that the buyer wants to close the contract (1616). The buyer can send the indication to the smart contract to initiate the closing of the contract. In some cases, the buyer can sign the indication or a message including the indication. For example, the buyer can send the indication via a message signed by the buyer using a buyer private key. The method then includes settling the contract via the smart contract based on a current value of the portfolio based on pricing data received from a trusted valuation entity (1618). The settling of the contract can be triggered by the indication that the buyer wants to close the contract. Thus, the buyer can control the settlement of the contract and request the settling when desired by the buyer. The current value of the portfolio can be calculated based on pricing data received from a trusted valuation entity. The contract can also be closed by other triggers besides just the buyer decision.

[0266] In some cases, a private key can hold the entire portfolio through the smart contract. Moreover, there is no

need for multiple wallets to manage multiple blockchain assets. Thus, the buyer can manage the portfolio containing various types of blockchain assets without having to maintain a respective wallet for each type of blockchain asset.

[0267] Settling the contract via the smart contract can include calculating, via the smart contracts, the current value of the portfolio and calculating a collateral amount and resulting proceeds to be sent to the buyer and the entity. The method can include sending a first resulting value from the resulting proceeds to the buyer and a second resulting value from the resulting proceeds to the entity. In some examples, the first resulting value can represent an original investment amount, plus a profit, plus an original excess collateral amount, and/or minus a commission. The second resulting value can include a difference from the original investment amount, minus a loss, plus the original excess collateral amount, and/or plus the commission.

[0268] In some cases, after receiving, at the smart contract, the indication that the buyer wants to close the contract, the entity can sell the portfolio of assets on an exchange. Moreover, in some cases, prior to receiving, at the smart contract, the indication that the buyer wants to close the contract, the method can include receiving a rebalancing request which indicates that the buyer wants to rebalance the portfolio of assets. The method can include receiving a new portfolio composition from the buyer based on the rebalancing request, calculating a new portfolio value and a new collateral value based on the new portfolio composition and data from the trusted valuation entity and accepting the new portfolio composition by the entity.

[0269] As part of this process, the method can include determining whether additional collateral is needed according to a collateral limit for the contract and receiving a confirmation of the new portfolio composition by the buyer and the entity. In some examples, the confirmation can be received via a buyer signed message from the buyer and an entity signed message from the entity. In some cases, the entity can buy or sell a mirror image of what the buyer bought and sold during a rebalancing of the contract.

[0270] FIG. 16B illustrates another example method for asset management in accordance with various aspects of the present technologies. The method includes receiving, from a buyer, an identification of a portfolio of assets associated with a blockchain-based smart contract (1620), receiving, from the buyer, an amount to invest in the portfolio of assets (1622) and determining a number of assets in the portfolio of assets based on one or more asset exchange rates and the amount to invest received from the buyer, to yield a composition of the portfolio of assets (1624).

[0271] In response to receiving, from the buyer, a confirmation for the composition of the portfolio of assets, the method includes calculating a cost of the portfolio of assets based on the composition of the portfolio of assets (1626), determining a portfolio contract based on the cost of the portfolio of assets (1628), receiving, from the buyer, a buyer acceptance of the portfolio contract, the buyer acceptance comprising the amount to invest and excess collateral (1630), identifying an entity acceptance of the portfolio contract in response to receiving, at the blockchain-based smart contract, an entity amount associated with the portfolio contract (1632) and receiving, at the blockchain-based smart contract, an indication that the buyer has requested to close the portfolio contract (1634).

[0272] In response to the indication that the buyer has requested to close the portfolio contract, the method can include settling the portfolio contract via the blockchain-based smart contract based on a current value of the portfolio of assets, the current value of the portfolio of assets being based on pricing data from a trusted valuation entity (1636). The portfolio of assets can include blockchain assets to yield a portfolio of blockchain assets. The identification of the portfolio of blockchain assets can include a respective percentage of each type of blockchain asset to enter into the portfolio of blockchain assets. The amount to invest can include an amount of at least one of Bitcoins, Ethereum, a cryptocurrency, and another blockchain asset. The amount to invest can also include a first amount in a currency or a second amount representing a physical asset. The acceptance can be associated with an entity, the entity being a seller of the portfolio of assets. Settling the portfolio contract can include calculating, via the blockchain-based smart contract, the current value of the portfolio of assets, a collateral amount, and resulting proceeds to be sent to the buyer and an entity associated with the entity acceptance. The resulting proceeds can include a first resulting value and a second resulting value. In this regard, the method can further include sending the first resulting value to the buyer and the second resulting value to the entity, the first resulting value including an original investment amount, plus a profit, plus an original excess collateral amount, and minus a commission, and the second resulting value including a difference from the original investment amount, minus a loss, plus the original excess collateral amount plus the commission.

[0273] The method can further include, after receiving the indication that the buyer has requested to close the portfolio contract, selling the portfolio of assets on an exchange. The method can also include, prior to receiving the indication that the buyer has requested to close the portfolio contract, receiving a rebalancing request indicating that the buyer has requested to rebalance the portfolio of assets, receiving a new portfolio composition from the buyer based on the rebalancing request, calculating a new portfolio value and a new collateral value based on the new portfolio composition and data from the trusted valuation entity, accepting the new portfolio composition by an entity associated with the entity acceptance of the portfolio contract, determining whether additional collateral is needed according to a collateral limit for the portfolio contract and receiving a second confirmation of the new portfolio composition by the buyer and the entity via a buyer signed message from the buyer and an entity signed message from the entity.

[0274] The method can further include providing, to the entity, a rebalancing fee prior to the accepting of the new portfolio composition by the entity. The entity can buy or sell a mirror image of assets bought and sold by the buyer during a rebalancing of the portfolio contract.

[0275] The indication that the buyer wants to close the contract can be received via a signed message from a buyer private key, and wherein the entity acceptance is received along with entity signature keys comprising private keys of a sending address of an entity associated with the entity acceptance.

[0276] The disclosure turns to a discussion of various security aspects of the application and approaches herein, an example history of attacks on decentralized approaches and example remedies and improvements included herein.

[0277] The disclosure will reference code documentation as well as information about smart contract security and best practices. After the well-known attack on the decentralized autonomous organization (DAO) on the Ethereum block chain in June 2016, many Ethereum users both old and new will have an increased skepticism about the security of decentralized applications built using smart contracts. Smart Contract technology is in its early days and the DAO hack, although unfortunate and difficult, was a wake-up call for smart contract developers to approach application design from a new perspective. It also showed that the underlying protocol did not experience a game-ending failure during this time of distress; an observation that should be comforting to those making investments in this platform.

[0278] In many ways the product and technology disclosed herein (PRISM) is the opposite of the DAO. The DAO had a very large amount of Ether in a single contract; whereas in PRISM, the product can hold small amounts of Ether (1 k to 50 k USD) in a single contract. In the DAO, there were thousands of individuals authorized to interact with a single smart contract; in contrast, with PRISM, there are two individuals (the buyer and the seller) and the oracle that are authorized to interact with each contract (and there will likely be thousands of contracts rather than just one). In the DAO, the contract was supposed to have a considerable life span (many years); whereas in PRISM, the product for each contract can be short-lived and disposable (e.g., from weeks to months for example). These considerations and others create a significantly different risk profile for the PRISM product when compared to the DAO. The risk profile and risk management approaches herein provide significant security checks and safeguards.

[0279] In one example remedy, a risk analysis can be applied to a contract in which steps are taken over time as a risk of a DAO type of attack increases to encourage completion of the contract with increasing intensity. For example, reminders could be sent to the seller with single click types of interactions to encourage the seller to close out the contract. Initial reminders could include more interactions to complete the process (such as three clicks in the security code) and later prompts to provide a more simplified interaction to close out the contract inasmuch as the risk factor has increased to a certain threshold.

[0280] The disclosure next addresses a risk assessment framework and methodology for decentralized applications. In this security analysis, the disclosure will separate the different sources of risk along the technology stack based on the root causes of those risks. This will help develop a framework for identifying, discussing, and managing these risks throughout the document. There are risks associated with using the Solidity Language. This includes risks derived from known and unknown bugs, faults, and/or issues with the Solidity programming language.

[0281] For example, it is the goal of PRISM to reduce Solidity risks by changing the code-base to reflect new smart contract development. There is a risk associated with using the Ethereum blockchain. This is the risk derived from the attributes of the Ethereum blockchain itself. This includes potential attacks on the protocol, hard-forks, community disagreements, consensus algorithm weaknesses, and more. There is a risk associated with Crypto Primitives. This is the risk derived from potential weaknesses in the fundamental hard-cryptographic primitives that secure such protocols. Things like any known or unknown weaknesses/attacks/

constraints in SHA256, ECDSA, TLS, etc. There is a risk associated with Business and Operations. This is the risk derived from the interaction of a centralized organization with the PRISM application and service. There's also risk associated with markets and volatility. This is the risk derived from rapid changes in the market value of either a single digital asset or a portfolio of digital assets. This risk is derived from the unpredictable nature of blockchain asset trading and includes the aforementioned Black Swan risks. Further risks relate to an Exchange Custodial risk. This is the risk derived from the observation that almost every exchange in the blockchain asset trading world is largely a centralized custodian of funds with almost no transparency into their solvency, audits, operational processes, etc. Finally, there is an Application Architecture Risk. This is the risk derived from how the application is fundamentally architected in terms of what actions parties are authorized to take via contract rules. For example, in PRISM, if the leader is authorized to change the settlement address of the follower's portfolio, this could create a security risk and vulnerability in the application.

[0282] The application next addresses remedies for some of these risks. The disclosure will begin with the Application Architecture Risk. One of the most common attacks results when an entity managing the contracts assumes the role of both portfolio seller and oracle. Creating "trustless" oracles is a topic in Ethereum and the overall discussion of smart contracts, so this discussion goes hand-in-hand. In a configuration where the managing entity is both the seller and the oracle, there are a number of attacks which could be launched by either a nefarious party or a hacker that gains control over the entity's private keys to the portfolio seller contract or the oracle. Below is an example attack which could be launched.

[0283] A fraudulent settlement attack which would push false data to the oracle is possible. In the case of a nefarious party, when the portfolio buyer asks to settle a contract, the nefarious party would push erroneous data to the oracle (like coin prices equal to zero) so that the portfolio value is zero, and subsequently all the collateral gets pushed to the nefarious party (the portfolio seller address in the contract). This equates to nefarious party stealing the user's funds.

[0284] The minimum viable product (MVP) was constructed under the assumption that since token pricing data is so public and transparent that if the nefarious party did this attack, it would be detected very quickly, and subsequently word would spread quickly, and consequently the nefarious party's business would be ruined. As previously noted, in the approaches herein, only the portfolio buyer can initiate settlement and so the nefarious party would have to wait for each portfolio buyer to ask to settle the contract before pushing false data. Such an attack would be detected quickly. In this scenario, the nefarious party could deceive a handful of customers and earn double their initial collateral deposit, but subsequently discourage any future users of the product. With smaller portfolios ranging from 1 k to 50 k dollars or ETH, the expected value of this attack is relatively low considering the upfront investment by the management entity to build the product in the first place (larger than multiples of 50 k). A nefarious party would be much wiser attempting to settle all outstanding contracts simultaneously based upon fraudulent settlement data rather than a handful of customers one-by-one as they come in to settle their portfolios. In this way, the nefarious party would be making

“one big grab” for all users’ funds existing in all outstanding contracts simultaneously. However, with the approaches herein, the nefarious party cannot settle all outstanding contracts simultaneously since only the buyer can initiate settlement.

[0285] In one example of the MVP (Prism.sol), the only time the portfolio seller is authorized to initiate settlement is when the collateral is all used; that is, the Current Portfolio Value (CPV) is zero.

[0286] This authorization is included to allow the management entity the ability to recoup its collateral in the event of a lethargic, unmotivated user who does not request settlement once their portfolio fails to produce returns (essentially, a portfolio dust account). In this case, the management entity is left unable to settle the contract and their collateral is subsequently tied up until the user finally initiates settlement of the contract by the life of the contract expires. However, a sellerSettle function can create the aforementioned attack, which is called The Big Grab Attack. It could be executed as follows: (1) Both the buyer and seller can call the portfolio.Value() function and this will trigger a request with the oracle for pricing data. At this point, the nefarious party would push data to the oracle with all coin prices equal to zero; (2) The contract would then read the value of the portfolio as “zero” based on the nefarious party’s erroneous data, and subsequently this would authorize the management entity to call sellerSettle, a function that forces the contract into settlement; and (3) The contract would attempt to settle the contract by reading data from the oracle; erroneous data pushed by the nefarious party that values the portfolio at zero, thereby settling the contract in the portfolio seller’s favor and subsequently pushing all collateral to the portfolio seller.

[0287] In this way, the nefarious party could quickly force all outstanding contracts into fraudulent settlement based on fraudulent oracle data, thus the name, The Big Grab Attack. Since management entity will be valuing every contract at least once per day (by calling the portfolio.Value() function) then this attack could be executed somewhat quickly.

[0288] There are various solutions to this attack. The first is to remove the sellerSettle function while the management entity is the only oracle provider. Another solution is to secure the oracle so that its functioning is not solely dependent on the management entity. In some cases, the system can do both in order to increase the application security on both fronts. However, simply removing sellerSettle would defend against this attack, since the nefarious party would have to wait for the user to request settlement and this by its very nature does not happen all at once. Thus, the nefarious party would be back in the same situation where it would have to settle each customer one-by-one with erroneous data; again, a tactic which would be detected very quickly, thus reducing the expected value of such an attack (especially since the user will be able to “pause” the contract if they suspect fowl play). If for some reason the nefarious party tried to do this to every customer, then customers would pause their contract for up to x days and there would be a long line of angry users outside of the management entity’s office requesting “fair settlement” to get their money back.

[0289] Regarding the oracle, there is an entire section dedicated to how to implement a secure oracle that will significantly reduce the risk of the oracle outputting false data or getting hacked itself. If the sellerSettle function

removed, how than does the system avoid the languishing portfolios problem? Efficient collateral management is important to the management entity’s approach.

[0290] In one example, each user was required to send ‘excess collateral’ to the contract. In the example, it was 25 ETH when the initial portfolio value was 100 ETH, and therefore, the collateral ratio was 1.25. If a portfolio’s value goes to zero, then the contract will still be holding 25 ETH of excess collateral. Instead of allowing the seller to force settlement of the portfolio at that point in time, the system will allow the seller to take a time-based subscription fee on a daily or weekly basis whenever the contract is valued. For example, the management entity would charge 1% per month of the total value of the contract for however long the contract is open. This is represented in the sellerWithdraw function. When this “excess collateral” goes to zero, then the contract would settle automatically. This is represented as a conditional inside the sellerWithdraw function. By “automatically”, this disclosure means that when contracts are evaluated via the portfolio.Value() function, the system would add a conditional which checks the buyer’s excess collateral, and if that number equals zero, then the portfolio will settle based on the prevailing rates from the oracle.

[0291] By taking a fee on a weekly or daily basis, and with a collateral ratio of 1.25, the user would be motivated to only use the wallet for its trading, portfolio management, and follow-leader functions, rather than as a MyEtherWallet type solution. Additionally, if a user decided to let a contract “languish”, it would not matter to management entity since management entity is earning fees from that contract at the same rate as any other contract which may be more “active”. The other benefit of this fee structure is that it scales in proportion to the Total Notional Value of all Outstanding Contracts (TNVOC); important because management entity infrastructure costs also scale with the total notional value (i.e. more contracts=more collateral=more hedging=etc.). Again, the contract would simply settle automatically when all the buyer’s excess collateral was spent; there is no need to give the seller any additional powers or authorizations.

[0292] Thus, the system will not authorize the management entity to force settlement of the contract as it opens up The Big Grab Attack which increases custodial risks for the users. A time-based fee hard-coded into the contract can make the contract automatically settle, motivates the user to use the service for its main features, and aligns the management entities incentives and costs with the user’s likely behavior. Designing the oracle to produce correct results is further discussed below.

[0293] Next, this disclosure discusses an Attack Surface Mapping and Scenario Analysis. In this section, the disclosure looks at possible attack combinations for all parties that have influence over the contract, which include the portfolio buyer, the portfolio seller, and the oracle. For each party to the contract there are two states from which they act: they are either hacked or not hacked. By being hacked, it is assumed that they will be acting nefariously and so the system does not have to consider the situation where they are not hacked but have for some reason decided to act nefariously. For example, a specific user may not be hacked but may decide to try and cheat the management entity. Either way, if the user was hacked or not, they are acting in clear contradiction to how the contract is designed to operate. Likewise, in The Big Grab Attack, the management entity could claim to have been hacked but really just stole them

money themselves, either way, the contract was somehow tricked into not producing the expected results. For example, expected results whose calculation is rather mundane with zero-room for interpretation or subjectivity. For this reason, the following discussion is going to fold the “non-hacked, but nefarious” party into the term “hacked”.

[0294] Again, the different states can include: (1) The management entity is hacked: somehow the attacker gets a hold of the private keys associated with the management entity’s position as the portfolio seller in the contract; (2) The management entity is not hacked: everything is fine; (3) The user is the hacker: since the attacker can take out swaps anonymously with management entity, then they don’t need to hack a specific user to assume this position in the contract. If they hacked a user, they could simply request settlement and take the funds. The product can let the user manage their own private keys, and so that is the user’s own liability. Thus, this disclosure only needs to consider if the user is actually the attacker; (4) The user is not the hacker: everything is fine; (5) The oracle is hacked: If the oracle is compromised then the data will be false (not true data); and (6) The oracle is not hacked: this means the oracle is not compromised, everything is fine, and the data is true.

[0295] The above scenarios are discussed now in more detail. Assume the “Big Grab Attack” in which the management entity is hacked, the oracle is hacked and the user is not the hacker. The approach to this scenario was discussed above.

[0296] In another scenario, assume the “Hacker Big Grab Attack” in which the contract state is that the management entity is not hacked, the oracle is hacked, and the user is the hacker. This is a new type of big grab attack. The attack would be executed as follows: (1) Attacker takes out a large number of swaps with the management entity as the buyer; (2) Attacker hacks the oracle; (3) Attacker initiates settlement of the swaps as buyer; and (4) Attacker pushes false data to the oracle.

[0297] In this case, the swaps will settle based on false data (inflated coin values), and the attacker will gain all of the collateral, and management entity’s collateral will be stolen. To defend against this attack, the system needs to make sure the oracle is secure and has failsafes. To do this, a security stack is proposed for the oracle in line with our previous discussions, but with much greater detail and added solutions. The following is a summary of the solutions which will be discussed in the oracle design section: (1) A multi-sig and validator oracle: Hacker would need to acquire more than just one key, if not all keys (M of N keys); (2) An oracle that receives data from multiple exchanges: Hacker would need more than just one key, if not all keys (i.e. hack all exchange oracle keys simultaneously); (3) An oracle that receives data from Oraclize: Hacker would need to find a weakness in the Oraclize smart contract as well as the TLS Notary technology. (Oraclize is a Solidity smart contract ensemble designed to make it easy for people to set up their own oracles using the public APIs of any website); (4) An oracle that receives “crowdsourced” data-validation from a group of users: This is something which the disclosure will elaborate on in the oracle design section. It is possibly the most decentralized oracle seen in the industry to date. This solution also lends itself to token issuance; and/or (5) Many of these layers and solutions can be combined. For example, one concept proposed is combining one through three in the product during its early stages of development. The goal is

actually to push as much risk as possible to the oracle and design the product to inherently defend against any user or management entity centric attacks. This is an object behind many smart contracts decentralized applications.

[0298] The above introduction to possible oracle solutions relates to oracle security in this attack, but the disclosure also introduces the aforementioned “failsafes” which will also be discussed in greater detail later.

[0299] One failsafe allows both the buyer and the seller to be authorized to “pause” a contract for some amount of time. The amount of time could be 24 hours, 7 days, or one month, or any other amount of time short or long. It could even be all three, that is, each party can choose for what period they want the contract to “pause”. When a party pauses a contract, it deactivates contract functions like buyerWithdraw, or the oracle, etc. This prevents those functions from producing undesirable results during this period. The “pause” feature can be implemented by using a multi-signature approach, a joint approach were the buyer and the seller must both authorize the pause, or an individual approach. Other variations could be built into the process, such as the buyer and/or seller each being given one “free” opportunity the pause the contract for 24 hours. One party could pay an additional amount of money for additional “pause” rights or time. All such variations are included within this disclosure.

[0300] Another failsafe is that each contract will also likely have a “settlement period” wherein even if the settlement calculation goes through without issue (that is, the data is true and neither party is hacked) then there could be a certain time before each party is allowed to withdraw funds. On a first glance, this should likely depend on the value of the contract, e.g. 24 hour period for contracts whose settlement value is less than 5 k USD, 48 hours for contracts of 5-20 k USD, and 7 days for contracts of 20 k USD or higher. Other time frames can also be built in the contract as noted above, such as providing interactions that urge the buyer or the seller to complete the contract based on some parameter such as an increasing risk to being hacked based on one or more of how long the contract has existed, outside external factors such as political changes, time of year, personal data about the parties to the contract obtained through social media, predicted issues that have a higher probability of occurring in the near or distant future, and so forth.

[0301] Another failsafe is that both the oracle and each contract itself can have a “manual override” function. The override acts as “training wheels” until the application is “battle tested”. A manual override function essentially allows a small set of trusted individuals (like curators in DAO) to completely override the contract in a number of ways. One power will be to manually send the contract settlement values instead of the oracle (in case the oracle is hacked). This manual override also can have other authorizations.

[0302] Yet another failsafe relates to a deployment plan with this application which is to slowly ramp up both the maximum allowable value for each contract as well as the total outstanding value of all contracts. The system starts with small portfolios with a limited number of total portfolios/contracts that are outstanding/open. As the application continues to operate securely, the system will increase the allowable value of each contract as well as the total outstanding value of all contracts. This is in contrast to how the DAO operated in that it solicited massive funds essentially during its alpha phase.

[0303] Reverting back to the subject of this specific attack (Hacker Big Grab), an important point to make is that although management entity would lose its collateral, management entity users would not be directly affected unless they were trying to settle their contract at the exact same time that the attacker was executing the attack. If this was the case, then those users would also be served false data, but likely also get inflated coin prices as this would be how the attacker would get the collateral pushed to them, and thus the customers would actually be “happy” since their portfolio got settled in their favor, and the management entity would lose its collateral.

[0304] Another scenario is the “Poor Extortionist Attack” in which the contract state is that the management entity is not hacked, the oracle is hacked and the user is not the hacker. In this attack, the attacker wouldn’t immediately gain anything since they are not the management entity or the user. If the attacker has control over the oracle, then why wouldn’t they just take out a bunch of swaps with the management entity before hacking the oracle? (This approach would be similar to the Hacker Big Grab Attack). In reality, the hacker probably would, but what if the attacker is poor and doesn’t have the capital to post to the swaps before taking over the oracle? How else could they monetize their access to the oracle? These questions shall be addressed next.

[0305] In the above scenario, the system would not allow any contracts to settle by pushing false data to the oracle, like all zero values. All zero values would make all portfolios worthless (and hence the user would never ask to settle them). Thus, the attacker could extort the management entity users to get their own collateral back by making them send ethers; or, simply wreak havoc on the entire system for whatever reason. This is an attack with a different execution profile, but with deleterious results nonetheless.

[0306] The inverse of the above attack could be that the attacker decides to inflate all portfolios to high values (makes all coins worth 1M USD) wherein everyone would initiate settlement (free money) and the management entity would falsely lose all of its collateral to its users. In this case, the attacker would gain nothing, however all the users would make gains; the attacker would likely see herself as being some kind of Robin Hood. The defense of this attack again rests with the security of the oracle for which this disclosure touched on in the previous attack and which is addressed under the oracle design options portion.

[0307] Another attack scenario is the “Hot Contract Attack”. In this scenario, the contract state is that the management entity is hacked, the oracle is fine, and the user is the hacker. This attack is very similar to what all exchanges face with having “hot wallets”. Here is how it would be executed: (1) Attacker would take out a non-trivial number of swaps with management entity; (2) Attacker would gain control of the portfolio seller role in those swaps by hacking management entity servers; (3) Attacker would initiate settlement of those swaps from the buyer role; and (4) Attacker would get all the collateral since they have assumed both sides of the swap regardless of oracle settlement data.

[0308] In this attack, all non-hacker users would not be affected, but the management entity would lose all its collateral. The defense to this attack depends on operational and business centric rules designed to protect management entity contract keys. The management entity has identified

some features which can be programmed into the smart contract that will help protect the management entity in this attack. The discussion of the next attack will demonstrate how these features will help the management entity.

[0309] The scenario is the “Poor Vampire Attack”. The contract state is that the management entity is hacked, the oracle is fine, and the user is not the hacker. In the past attack, the user is the hacker, and so they would get all the collateral once they initiate settlement from the buyer role. In this version, the attacker has gained control over the management entity contract keys, but has not taken out a bunch of swaps with the management entity. The reason for this could be the same as in the Poor Extortionist Attack in that the attacker doesn’t have much capital to start with, and so she looks to monetize this access in another way.

[0310] In this attack, once the hacker steals the management entity contract keys, they would just sit and wait for each user to request settlement over time. The management entity would be powerless in this situation and the hacker would slowly drain the collateral out of each contract, like a Vampire, as each user went to settle their contract. However, management entity users would not be affected in this attack since they would get settled fairly as the oracle is fine.

[0311] To reduce the risk and value of this attack, this disclosure proposes that contracts include a new function designed to invalidate the Hot Keys registered in the contract as the portfolio seller role and revert it to a new address which it is called the Back Up Key(s). If the system suspects foul play, or if all server-side keys have been compromised, then with one simple call to this function, the system can quickly invalidate all active contract keys and revert the portfolio seller address to the Back Up Key. There could even be two layers here in that there could two or three Back Up Keys. The Back Up Key(s) could be generated and managed like a cold storage/contract key in that it is never really exposed to an attacker since it is generated offline, and only needs to sign a message in the event the system suspects a Hot Contract key has been compromised, which will hopefully be never.

[0312] This disclosure recommends a hack prevention process for any key holder authorized to push data to the oracle. In this way, no oracle participant should ever be able to reasonably claim that their oracle key was hacked for any meaningful length of time; all they will have to do is use the Back Up Key to invalidate the Hot Key (which in the case of the oracle will be trivial to detect since there will be several parties validating the data at each settlement).

[0313] In this configuration, the first time management entity notices that any settlement funds have been diverted out of their control, they can simply swap any single key—or all keys—long before other user’s request settlement. This should deter hackers since it is the equivalent of not being to do a “big grab” on a centralized exchanges “hot wallet” since the hacker would have to wait for each and every user to make a withdrawal request from the exchange to get their money. User withdrawals by their very nature do not happen all at once and so the attacker could wait a very long time to drain all the hot wallet funds. Once that first withdrawal/settlement appears compromised, then management entity can quickly invalidate all the “hot wallet keys” and replace them with fresh keys. Adding such a key replacement function is secure from a Solidity programming perspective.

[0314] This defense significantly reduces the risk of loss during the Hot Wallet Attack and the Poor Vampire Attack wherein management entity contract keys are hacked. Again, remember that in both of the attacks non-hacked PRISM users are not affected.

[0315] Another scenario is the “Business As Usual #1”. In this case, the contract state is that the management entity is not hacked, oracle is fine, and the user is the hacker. There is no application level attack here. The product is designed to protect management entity against a nefarious/hacked user and likewise the product is designed to protect the user against a nefarious/hacked management entity (with the aforementioned caveats).

[0316] Another scenario is the “Business As Usual #2”. In this case, the contract state is that the management entity is not hacked, oracle is fine, and the user is not the hacker. There is no issue in this case, everything is working fine, and the management entity operates business as usual.

[0317] Another scenario is the “Hell Fire and Brimstone” Attack. In this case, the contract state is that the management entity is hacked, oracle is hacked, and the user is the hacker. In this case, all security features have simultaneously failed, the attacker decidedly can do anything upon any party she so pleases. The last three scenarios above are provided for completeness in the attack scenarios as there are three parties to the contract, the portfolio buyer, the portfolio seller and the oracle, each of which can be either hacked or not hacked.

[0318] After making the design considerations mentioned above, the only attack which would hurt users of the disclosed marketplace (PRISM) are the first, third and eighth attacks above. With attack #1, the system can simply remove the sellerSettle function as discussed. At which point, this attack morphs into the third attack, in which just the oracle is hacked and the attacker extorts users to get “fair settlement” (Poor Extortionist Attack). If the last attack occurs, then it will be difficult. Thus, this is the attack which needs to be defended against. Securing the oracle is paramount to avoiding this attack and so a discussion of the oracle design and implementation is next.

[0319] In this section the disclosure will introduce and discuss the different oracle design options and see their effects on the oracle’s security. The first option with respect to the oracle design is a multi-validator oracle. A multi-signature technology can be implemented for the oracle. If so, for data to be accepted by the oracle as “truth”, it would need to be signed off by M of N parties to the transaction.

[0320] Having the oracle be multi-signature applies this level of security and comfort to the system. There is no altcoin exchange that can claim they have multi-sig for all their wallets. This is because it takes time and effort to build a multi-sig implementation for each and every coin. Thus, the present approach shifts that technical problem directly to securing the oracle data rather than the private keys of each coin. This alone is interesting from a sales and marketing perspective as multi-signature is a cryptographic concept and from the outset the system can claim to be “more secure” than any centralized exchange (based on the oracle security).

[0321] The implementation disclosed herein can implement a re-imagination of multi-signature concepts. In the present disclosure, the system can ask an entity, such as Bitgo (or any other entity), to not only be a signer but also validate the oracle data at the same time. This means that

Bitgo would have to run an instance of the CoinCap.io API, pulling the source data from the public APIs of each contributing exchange, running the CoinCap.io calculation methodology, and then pushing that data to the oracle.

[0322] In this configuration, Bitgo (or any entity performing this function) is both a key holder and a validator. The entity would not be blindly signing oracle data and would have to provide the right result and corresponding calculation proofs.

[0323] As noted above, the entity does not have to be Bitgo, and could be an array of trusted validators and signors all pulling data from the public APIs, producing the calculation results, and when everyone is in agreement, only then signing for the authenticity of this data for oracle use.

[0324] The multi-validator process could also be enforced by a single smart contract itself rather than having the key-derivation function be a multi-signature process. In a pure smart contract implementation, oracle data would not be validated until M of N key-holders signed off on its authenticity. The difference in these two approaches could be up for discussion based since in multi-signature the source of risk is a Crypto-Primitives Risk versus in the pure smart contract implementation the source of risk is Solidity Language Risk.

[0325] Some of these concepts apply in a pre-Bitfinex hack world, and since Bitfinex was using Bitgo, there are new product implementations with suggesting they could be a signor/validator. Until the system knows the source of the failure for the Bitfinex hack, Bitgo as a specifically suggested entity may not be an option. Pre-Bitfinex hack, most people likely agreed they would have been a potential good choice for this role.

[0326] A second option for developing the oracle is the multi-exchange oracle in which the exchanges push data. In this case, the oracle would contact the exchanges and ask them to push their trading data directly to the oracle. The oracle would only be authorized to receive data from certain Ethereum addresses, each of which would be controlled by a different exchange. The system can also authorize the oracle to receive data from other addresses from a group of trusted addresses. Each exchange would just push the data from their API directly to the oracle. An attacker would need to simultaneously compromise each of the exchanges oracle keys, and then simultaneously push false data to the oracle to execute the attacks previously discussed. Although this does increase the security of the oracle significantly, there are two important considerations which arise from this configuration.

[0327] The first is a business development and product-centric consideration. Implementing this type of oracle would require the cooperation of each exchange. They would have to take responsibility for managing their oracle key and then pushing the data to the oracle upon request. Many exchanges may do this to help build the ETH ecosystem, while some exchanges may see PRISM as a competitor for their small/mid-dollar retail clients. Successful exchanges have built liquidity through attracting all different types of clients including retail clients. Further, a sales and marketing pitch is definitely geared toward showing that the product has significantly less custodial risk than any exchange.

[0328] The second aspect is a modification on the Hacker Big Grab Attack which could be executed by a nefarious exchange instead of an independent attacker of the oracle.

Assume that after contacting all the exchanges that only one exchange agrees to push data to the oracle, e.g. Poloneix. Therefore, all of the oracle data is coming from them. Here is how an attack could execute: (1) Take out a bunch of swaps with management entity as a series of anonymous users; (2) Request settlement of those portfolios from the buyer role; (3) Push fake data to the oracle which erroneously inflates the value of those users' portfolios; (4) All the collateral would wrongly get pushed to the buyers; (5) The management entity then contacts the exchange asking "What's going on with the data?"; and (6) Exchange claims it was hacked, but quietly makes off with a fortune.

[0329] This attack is called the Nefarious Exchange Big Grab Attack. A solution to this attack is to have a diversity of exchanges pushing data to the oracle. However, this could create a bottleneck on the flexibility of the service since there are some coins which Poloneix does not trade. Conversely, there are some coins that only one exchange trades (e.g. Gatecoin for Augur and DGX IOUs). This attack could be executed against a single coin portfolio if that coin is only traded on one exchange. Another good solution to this problem reverts back to the process mentioned with Bitgo, in that Bitgo would be pulling the data from the exchanges public APIs. The main benefit here is that the exchange couldn't claim that its oracle keys were hacked. The only way they could manipulate the data at that point would be to orchestrate some type of flash-crash on their order book. This is far more difficult and has broader ramifications for the exchange since their own users will be severely affected by such a flash crash (whereas if they claimed a hack on the oracle, none of their customers would be affected). Further, if management entity or another party like Bitgo is pulling the data from those exchanges, then the system can do sanity checks on the data before signing its authenticity to the oracle. For example, if the system observes a "flash crash" happening, it simply "pause" transactions by not signing the oracle data until the "flash crash" is over. The system would want to have such sanity checks on trading data for market risk centric reasons. In the traditional finance world, such "trading pauses" during a flash crash or ultra high volatility period are called "circuit breakers". Note that a circuit breaker approach with management entity authorized to pause settlement would also defend against a Big Grab Attack where the oracle was compromised by an outside attacker (since they would try and inflate prices).

[0330] Although the approaches herein can implement a process where exchanges push data to the oracle from their own oracle keys, some configurations can allow an entity to pull the data from public APIs for more security to the management entity. Additionally, the robustness of the overall oracle (number of coins that can be offered, number of data providers, etc.) will not be dependent on getting each and every exchanges "approval" to use their data for the oracle.

[0331] The disclosure now turns to a discussion on Oraclize, which would allow a management entity to pull data from exchanges public APIs without their permission in a "provably honest" way. Another option is to use the Oraclize Technology and Service. Oraclize is a Solidity smart contract ensemble designed to help people set up their own oracles using the public APIs of any website. It does this through a suite of smart contracts, a pseudo-centralized service which they provide, and a technology called TLS Notary which is designed to make their centralized pro-

cesses "provably honest". A TLS Notary allows a client to provide evidence to a third party auditory that certain web traffic occurred between himself and a server. The evidence is irrefutable as long as the auditor trusts the server's public key. Essentially, Oraclize queries a target API and then pushes the data to the designated oracle for the application.

[0332] The following is an example for using Oraclize in the present application. A CoinCap.io instance is set up with Oraclize in which Oraclize, upon every request, pulls data from all the public APIs of all the exchanges and then computes the resulting values (using the CoinCap.io method). Next, Oraclize pushes the resulting data to the portfolio contracts oracle (that will be used for settlement) and at the same time includes two things: (1) A TLS Notary proof and (2) The location of the source data and results in the interplanetary file system (IPFS). All of this is done through a smart contract except for when the data must be queried from the exchanges public API, since smart contracts cannot reach out to APIs. Thus, the Oraclize centralized server makes the query, which is where TLS Notary operates.

[0333] TLS Notary can work by generating a cryptographic proof during the TLS Handshake process which allows one to prove to any third party that the data delivered from the server's HMAC key has not been manipulated by anyone. Therefore, the user is able to prove that the data sent from that server has integrity. This is useful because the major risk with using Oraclize is that it could manipulate the data between when it queries it from the public API of an exchange and pushes it to the oracle. This would open the process up to a Big Grab Attack since Oraclize could all of sudden push false data to the oracle and then settle a bunch of swaps it took out against the management entity. However, with the TLS Notary proof, the management entity can mathematically verify that they have not manipulated the data when it was sent from the exchange API.

[0334] All the management entity has to do once it receives the data is run the verification on the TLS proof and it would know that Oraclize did not change anything. Of course, the system can also check their result with the CoinCap.io results and make sure they match. Oraclize stores the source data, the result, and the TLS Proof in the interplanetary file system so that anyone can pull up the file and run the validation independently without having to request the files from Oraclize.

[0335] Another option relates to a Browser Wallet Validation Tool in which the users derive data. Another interesting way to validate the CoinCapi.io data is to have the user re-derive it for every settlement. This could be done by the user running an instance of the CoinCap.io calculation methodology but pulling the source data themselves. This could be implemented as a browser tool in MyEtherWallet or as a desktop tool connected to Mist/MetaMask. It would work as follows: (1) A user would initiate settlement from the buyer role; (2) The tool would then pull the public API data and calculate the results; and (3) The tools sends that result as a signed message to the oracle. A benefit of this approach is that this solution is attached to the private key of the user's public address in the contract. If every user was pulling and validating trading data and then sending the results in a signed message, that would be close to a fully decentralized oracle.

[0336] Another aspect of this option is that it allows non-technical users to execute it without their knowledge.

For example, if the system were running it from MyEther-Wallet, then the process would just occur in-browser through JavaScript calls to the public APIs of each exchange. Of course, the big question is what if the user lies about data (changes the APIs results) for whatever reason. What can be learned in the crypto-world to disincentivize anonymous users from lying about transaction data? The disincentivization can be in requiring a proof of work, proof of stake, or another consensus algorithm.

[0337] The following is a further possible oracle implementation. Considering the options and trade-offs, the following is an example embodiment for the oracle. In one aspect, 50-100 user contracts with maximum contract value of 1 ETH per contract could be implemented. In this initial case, the system would just use CoinCap.io API that will push data directly to oracle. In a later version, 100-200 user contracts are initiated but with a maximum contract value of 100 ETH and with a Total Outstanding Notional Value of 100 k-300 k USD. In this case, the system would add Oraclize CoinCap.io data validation process and provide two contributors to oracle the management entity (such as a management entity) and Oraclize. Thus, a tiered approach to the data validation process could be implemented based on one or more factors in the portfolio of contracts.

[0338] Another version could serve a maximum of 200-400 customers with a maximum contract value of 500 ETH and with a Total Outstanding Notional of \$1M-2M USD. The system could add Bitgo/EtherLi multi-sig to the validation process, two signers/validators (2 of 2 multi-sig) and one contributor (Oraclize) to oracle: management entity, Oraclize, and Bitgo/EtherLi. The system would add new sellers to the equation to share risk with management entity for collateralizing portfolios, and let sellers be signatory (bump multi-sig) and sellers would have a maximum of 250 k they can use to collateralize (target of 2 sellers onboarded).

[0339] Integrating Oraclize will work since their service is specifically designed to make setup of an oracle easy, and this is one of the reasons why it is recommended to integrate them first. Having Bitgo be a party in the transaction is good since: (1) The multi-sig tech is established; (2) the multi-sig tech is pretty simple and EtherLi is functional today, so integration time and effort would be low; and (3) just the sales pitch of “crypto-asset portfolios secured by multi-sig/validation model” can make the management entity more secure than any exchange, never mind that Oraclize is integrated too. From Bitgo’s perspective, this could be a lucrative new business model: validating data, computations, and being an impartial signatory to all types of different oracles.

[0340] A user validation tool can also be on the list to combat users forging data and trying to trick the system. These decentralized “data validation” processes combined with “simple contracts” are the likely near-term future of Ethereum until the protocol has years of debugging and testing similar to Bitcoin’s protocol progression. A startup called Truebit by Dr. Christian Reitwiessner is an off-chain data validation tool/process, which publishes proofs to “simple contracts” so that more complex computational operations can be securely executed in Ethereum’s framework (rather than coding all the logic to be executed on-chain). Such technologies could be applied here and are incorporated herein by reference.

[0341] The following is one possible way in which a settlement can occur in this configuration: (1) A user requests settlement of their contract; (2) The contract fires a message to Oraclize requesting a settlement result; (3) Oraclize pulls the data, computes the result, and sends it with proofs to the oracle; (4) The oracle waits for management entity and Bitgo to validate the results and the proof; (5) The management entity and Bitgo both sign the result and the proof; (6) The oracle takes the result and sends it to the contract; and (7) Contract makes the settlement based on the validation oracle data. There are other implementations and ways to address this as well and the above provides an example settlement approach.

[0342] There are other configurations that can be included and applied as part of this disclosure and application. These options include: (1) Smartcontract.com, (2) Feedbase, (3) PriceGeth, (4) Microtick, etc. Smartcontract.com acts like Oraclize by executing calls to public APIs and then pushing the data to an application oracle. In the present configuration, they could be another “contributor” and also a back-up in the case that Oraclize fails. They are also working with Brave New Coin, which is a data aggregator of crypto-asset exchange prices and that is the entity that will be pushing data to the oracle. Feedbase is a simple tool that allows people to push data to the blockchain from geth in a standardized way. This could be employed in the user validation process. PriceGeth is a smart contract and python file/server that is fetching poloneix data from the public API and then writing it to the ETH blockchain. This technology is leveraged by the present system based on the above discussion. Microtick is an implementation of Vitalik’s Schelling Coin paper using the “betting” method as opposed to the consensus or sampling method. It is a browser implementation. The details of each of these concepts that can be gained through access to their website is incorporated herein by reference. A number of oracle options can be used for increasing the security of any oracle.

[0343] Next, the disclosure addresses the attack surface for the oracle configuration and scenarios. In the recommended configuration, there are two signers/validators and one contributor. This disclosure does not consider Oraclize to be a “signer” because they are only contributing the source data and calculation results to the oracle but they will not have any signing power for authorizing the data to be used in the settlement of contracts. Rather, it is the job of management entity and Bitgo to validate Oraclize data and then sign for its authenticity for the transactions to go through. While Bitgo is the example, the scenarios could apply to any entity performing a similar function. The following provides a “name” of a particular scenario, a “state” of that scenario and a “result”.

[0344] Name 1: Business as Usual.

[0345] State 1: Oraclize is not hacked, management entity is not hacked, Bitgo is not hacked.

[0346] Result 1: All data matches, management entity and Bitgo sign transaction, data used for settlement.

[0347] Name 2: Paused by management entity.

[0348] State 2: Oraclize is not hacked, management entity is hacked, Bitgo is not hacked.

[0349] Result 2: Management entity data does not match. Transaction is paused. Management entity invalidates its primary key with the oracle using the back-up key. Management entity signs data with back-up key and transaction goes through.

[0350] Name 3: Paused by Bitgo.

[0351] State 3: Oraclize is not hacked, management entity is not hacked, Bitgo is hacked.

[0352] Result 3: Bitgo data does not match. Transaction is paused. Bitgo invalidates its primary key with the oracle using the back-up key. Bitgo signs data with back-up key and transaction goes through.

[0353] Name 4: Paused by Signers and oracle Switched #1.

[0354] State 4: Oraclize is not hacked, management entity is hacked, Bitgo is hacked.

[0355] Result 4: The management entity and Bitgo data does not match Oraclize. Transaction is paused. Bitgo and management entity invalidate their primary keys with the oracle using the back-up key. Both signs the transaction using back up keys and data is used for settlement.

[0356] Name 5: Paused by Signers and oracle Switched #2.

[0357] State 5: Oraclize is hacked, management entity is not hacked, Bitgo is not hacked.

[0358] Result 5: The management entity and Bitgo data does not match Oraclize. Transaction is paused. If Oraclize protocol is cryptographically broken, then the system has to remove them as a data source from the oracle.

[0359] How would removing them from the data source be accomplished? This disclosure introduces a function called "oracle override". This function can be called by two of the three parties. The keys for the oracle override function are separate from the primary and backup signing keys for data validation, and so the override keys will always be held in cold storage until this event occurs (hopefully never). To repair the oracle in this situation, management entity and Bitgo call the oracle override function and appoint management entity as the new oracle data source. Both management entity and Bitgo sign the new data and the transaction goes through.

[0360] Name 6: Paused by Bitgo and oracle Switched.

[0361] State 6: Oraclize is hacked, management entity is hacked, Bitgo is not hacked.

[0362] Result 6: The management entity and Oraclize data does not match Bitgo data. The management entity invalidates primary oracle keys with back up keys. If Oraclize protocol is broken, the management entity and Bitgo retrieve Override Keys from cold storage and authorize the oracle to switch to CoinCap.io data. The management entity and Bitgo sign correct data and transaction goes through.

[0363] Name 7: Paused by management entity and oracle Switched.

[0364] State 7: Oraclize is hacked, management entity is not hacked, Bitgo is hacked.

[0365] Result 7: Bitgo and Oraclize data does not match management entity. Bitgo invalidates primary oracle keys with back up keys. If Oraclize protocol is broken, management entity and Bitgo retrieve Override Keys from cold storage and authorize the oracle to switch to CoinCap.io data. The management entity and Bitgo sign correct data and transaction goes through.

[0366] Name 8: Hell Fire and Brimstone, Again.

[0367] State 8: Oraclize is hacked, management entity is hacked, Bitgo is hacked.

[0368] Result 8: Attacker has broken the protocol and can do whatever she wants. The novelty here is to make the total expected value of a hack less than the effort and combined expertise required to execute the hack; i.e. the Total Notional

Value of Outstanding Contracts is 1M USD, not 214M USD, like the DAO. However, this level oracle security could handle up to 25M USD easily. Bitgo is already securing more than that with its Bitcoin solutions.

[0369] Since there are three parties to the oracle: management entity, Bitgo, and Oraclize, and two possible states per party (hacked or not hacked), then the total attack surface is $2*2*2=8$ scenarios. Thus, all the scenarios are covered.

[0370] Next, the application addresses manual override, speed bumps, and training wheels. In the last discussion of the oracle, the ability to "pause" the contract by either of the two signing parties was introduced should it appear that something is amiss. The oracle override function was also introduced, a function which acts as an escape hatch should the Oraclize service get hacked. Lastly, an off-chain multi-validator model was introduced in which three different parties made calls to the public APIs of the source data and then derived the results by running an instance of the CoinCap.io calculation methodology.

[0371] The following extends these same concepts directly to each and every contract itself. As mentioned above, this disclosure must also develop a plan to deal with Solidity Language Risk both known and unknown. In the case of unknown bugs, the best that can be done is: (1) Add the ability to pause a contract for a certain amount of time. This equates to freezing the contracts state and select functions for all parties; (2) Add a manual override function which allows a contract to be settled based on manual inputs from a group of trusted validators; and (3) Open-source the contract validation and monitoring tools which management entity has developed for their own internal purposes. In this way, anyone can use them to validate that contracts are behaving as planned.

[0372] Expanding on the third point, one could implement this entire application via off-chain logic and several validators. That is, one could remove almost all the logic currently coded in the Solidity files and instead use an escrow contract that sends funds to each party based on the collective agreement of the settlement amounts by the seller, the buyer, and any number of third-party Arbitrators/Validators.

[0373] Instead, the present disclosure proposes to use a Training Wheels configuration where much of the logic stays on-chain but a certain set of validators/signers will be able to manually override/settle, pause, or cancel a contract if it's not behaving as expected. These validators/signers could be the same as in the oracle (Bitgo), or another set of publicly identified arbitrators (Roger Ver, for example).

[0374] This configuration can be thought of as a type of "Training Wheels" configuration for the application because more code is left on-chain but have trusted parties who can jump to the rescue the contract if things do not work. As mentioned previously, the goal with one embodiment of this disclosure is for it to have significantly "less trust" than the current alternatives in the marketplace, all of which are highly-centralized blockchain asset exchanges. With all of the risk placed on the oracle, and with an open and transparent oracle process that involves three separate parties rather than one (like current exchanges) big improvements to the fully centralized model that currently dominates the marketplace have been presented.

[0375] An example embodiment of the concepts disclosed herein includes a computer-readable storage device. The computer-readable storage device is a man-made physical

device such as RAM, ROM, a hard drive, optical drive, or any other device that can store instructions which, when executed by a processor, can cause the processor to perform operations including any one or more of the steps or processes disclosed herein. The computer-readable storage device excludes signals per se and the like.

[0376] The present examples are to be considered as illustrative and not restrictive, and the examples is not to be limited to the details given herein, but may be modified within the scope of the appended claims. It is further noted that any feature of any example or any embodiment may be mixed with any other feature disclosed herein. Any method embodiment which includes a series of steps may also include, as one aspect, only one or two of the listed steps. The order of the listed steps may also be modified and performed in any order unless explicitly required.

[0377] Claim language reciting “at least one of” a set indicates that one member of the set or multiple members of the set satisfy the claim. For example, claim language reciting “at least one of A and B” means A, B, or A and B. Use of the terms “at least one of” to describe a list of items separated by the term “and” should not be interpreted to mean that the list requires one of each item in the list, but rather that the list includes at least one item in the list, which can be any item or number of items in the list, such as the full set of items in the list, a subset of items in the list, or a single item in the list.

[0378] Statements of the Disclosure Include:

[0379] Statement 1: A method comprising: receiving, from with a user interface associated with a user, one or more parameters associated with a creation of a customized smart contract for a multi-asset portfolio; authenticating one or more parameters via a public key or a private key associated with the user; and deploying the customized smart contract onto a blockchain.

[0380] Statement 2: The method according to Statement 1, further comprising: generating the customized smart contract to implement the one or more parameters, wherein the customized smart contract is written in a language having a syntax designed to compile code for the Ethereum Virtual Machine (EVM).

[0381] Statement 3: The method according to Statement 1 or Statement 2, further comprising: implementing the customized smart contract on the EVM.

[0382] Statement 4: The method according to any one of Statements 1 to 3, wherein the blockchain is a public blockchain, and wherein deploying the customized smart contract comprises deploying the customized smart contract on the public blockchain.

[0383] Statement 5: The method according to any one of Statements 1 to 4, wherein the customized smart contract runs without a custodial risk.

[0384] Statement 6: The method according to any one of Statements 1 to 5, wherein the customized smart contract is between a first party and a second party and wherein no third party holds custody of any assets associated with the customized smart contract.

[0385] Statement 7: The method according to any one of Statements 1 to 6, wherein the one or more parameters are associated with one or more of auctions, wallets, real estate transactions, stock trades, altcoin trades, crowdfunding, contracts, legal services, and currencies.

[0386] Statement 8: The method according to any one of Statements 1 to 7, wherein the customized smart contract is coded, stored and replicated in a distributed platform.

[0387] Statement 9: A system comprising one or more processors and at least one computer-readable storage device having stored therein instructions which, when executed by the one or more processors, cause the one or more processors to perform a method according to any one of Statements 1 to 8.

[0388] Statement 10: At least one computer-readable storage device having stored therein instructions which, when executed by the one or more processors, cause the one or more processors to perform a method according to any one of Statements 1 to 8.

[0389] Statement 11: A system comprising means for performing a method according to any one of Statements.

1. A method of publishing a leaderboard of blockchain portfolio swap smart contracts, the method comprising:

generating a list of blockchain portfolio swap smart contracts, each representing a portfolio of blockchain assets swapped for exposure against a funding blockchain asset between a buyer and a seller, the blockchain portfolio swap smart contracts binding the buyer and the seller to share a payout in the funding blockchain asset based on a value of the portfolio of blockchain assets upon closing of the portfolio swap smart contract;

analyzing the list of blockchain portfolio swap smart contracts to yield at least one performance metric and to yield status information regarding each blockchain portfolio swap smart contract;

sorting the list of blockchain portfolio swap smart contracts based on the at least one performance metric;

receiving a request from a potential buyer for the list of blockchain portfolio swap smart contracts; and

publishing the list of blockchain portfolio swap smart contracts to the potential buyer, each blockchain portfolio swap smart contract displaying the status information and performance metric associated therewith.

2. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, further comprising:

receiving, from the potential buyer, a request to clone a leader blockchain portfolio swap smart contract in the list to yield a follower blockchain portfolio swap smart contract; and

deploying the follower blockchain portfolio swap smart contract to a blockchain of the funding blockchain asset, the follower blockchain portfolio swap smart contract being programmed to periodically rebalance to a blockchain portfolio asset allocation based on a current blockchain portfolio asset allocation of the leader blockchain portfolio swap smart contract.

3. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, further comprising:

publishing the list of portfolio swap smart contracts to potential sellers including an amount of the funding blockchain asset sufficient to fund each blockchain portfolio swap smart contract and a blockchain address of each follower blockchain portfolio swap smart contract;

receiving an indication of agreement to fund one of the portfolio swap smart contracts; and

revising the list published to potential buyers to indicate that at least one seller is willing to fund the one of the portfolio swap smart contracts.

4. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, wherein the list of blockchain portfolio swap smart contract published to the potential buyer includes an indication of whether any seller has agreed to fund a follower blockchain portfolio swap smart contracts purchased as clones of a blockchain portfolio swap smart contracts on the list.

5. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, wherein the operation that generates the list of blockchain portfolio swap smart contracts includes scanning a blockchain of the funding asset and not receiving elements of the list from any third-party.

6. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 5, wherein the request from the potential buyer to clone the leader blockchain portfolio swap smart contract includes a request for the follower blockchain portfolio swap smart contract to periodically rebalance to a blockchain asset portfolio that is only a portion of a rebalance to the blockchain asset portfolio of the leader blockchain portfolio swap smart contract.

7. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, wherein the portfolio of blockchain assets includes assets pegged to the value of one or more of auctions, wallets, real estate transactions, stock trades, altcoin trades, crowdfunding, contracts, legal services, and currencies.

8. The method of publishing a leaderboard of blockchain portfolio swap smart contracts of claim 1, wherein the operation that publishes the list of blockchain portfolio swap smart contracts is a publication to a social media feed.

9. A method of cloning a leader blockchain portfolio swap smart contract to yield a follower blockchain portfolio swap smart contract comprising:

requesting, from a blockchain portfolio swap smart contract marketplace, a leaderboard list of blockchain portfolio swap smart contracts, the blockchain portfolio swap smart contracts in the list each representing a portfolio of blockchain assets to swap for exposure against a funding blockchain asset between a buyer and a seller, the blockchain portfolio swap smart contracts binding the buyer and the seller to share a payout in the funding blockchain asset based on a value of the portfolio of blockchain assets upon closing of the portfolio swap smart contract;

receiving, from the blockchain portfolio swap smart contract marketplace, the list of blockchain portfolio swap smart contracts sorted according to a performance metric;

requesting creation of a new follower blockchain portfolio swap smart contract cloning a blockchain portfolio swap smart contract in the leaderboard list such that the new follower blockchain portfolio swap smart contract is programmed to periodically rebalance to a blockchain portfolio asset allocation based on a current blockchain portfolio asset allocation of the leader blockchain portfolio swap smart contract;

receiving a blockchain portfolio swap smart contract address of the new follower blockchain portfolio swap smart contract;

broadcasting a buyer funding transaction to the smart contract address of the new follower blockchain portfolio swap smart contract on the blockchain of the funding asset; and

receiving confirmation that a seller has deposited a seller funding transaction to the smart contract address of the new follower blockchain portfolio swap smart contract in an amount equal to the buyer funding transaction, thus providing opposite exposure to the buyer and the seller with respect to the portfolio of blockchain assets.

10. The method of claim 9, further comprising:

receiving, from the blockchain portfolio swap smart contract marketplace, in response to the operation requesting creation of the new follower blockchain portfolio swap smart contract, one or more options for the periodic rebalance of the portfolio of blockchain assets of the follower blockchain portfolio swap smart contract that differ from a straight mirroring of the portfolio of blockchain assets of the leader blockchain portfolio swap smart contract.

11. The method of claim 10, wherein the options for the periodic rebalance of the portfolio of blockchain assets of the follower blockchain portfolio swap smart contract include a time delay between rebalancing of the leader blockchain portfolio swap smart contract until the follower blockchain portfolio swap smart contract rebalances during which the buyer or the seller of the follower blockchain portfolio swap smart contract can cancel rebalancing by sending a transaction to the address of the follower blockchain portfolio swap smart contract on a blockchain of the funding asset.

12. The method of claim 10, wherein the options for the periodic rebalance of the portfolio of blockchain assets of the follower blockchain portfolio swap smart contract include approval of the buyer of the follower blockchain portfolio swap smart contract via transaction sent to the address of the follower blockchain portfolio swap smart contract before rebalancing can occur.

13. The method of claim 10, wherein the operation that received one or more options for the periodic rebalance of the portfolio of blockchain assets of the follower blockchain portfolio swap smart contract includes receiving market research on the portfolio of blockchain assets.

14. The method of claim 13, wherein the new follower portfolio of blockchain assets is programmed to periodically rebalance to only a blockchain portfolio asset allocation that is only a portion of the blockchain portfolio asset allocation of the leader blockchain portfolio swap smart contract.

15. The method of claim 9, further comprising:

requesting creation of cold wallet follower blockchain portfolio swap smart contract cloning the new follower blockchain portfolio swap smart contract with a blockchain asset portfolio value significantly greater than the new follower blockchain portfolio swap smart contract.

16. The method of claim 9, wherein leader blockchain portfolio swap smart contract and the new follower blockchain portfolio swap smart contract are both controlled by the buyer, the leader blockchain portfolio swap smart contract being a hot wallet contract and the follower blockchain portfolio swap smart contract being a cold wallet contract with a blockchain asset portfolio significantly larger than the hot wallet contract such that rebalancing of the cold wallet contract can be done with the signing key to the hot wallet contract without exposing the hot wallet key to theft.

17. A method comprising:
requesting, from a blockchain portfolio swap smart contract marketplace, a list of blockchain portfolio swap smart contracts for which a potential buyer has indicated a desire to buy, each representing a portfolio of blockchain assets swapped for exposure against a funding blockchain asset between a buyer and a seller, the blockchain portfolio swap smart contracts binding the buyer and the seller to share a payout in the funding blockchain asset based on a value of the portfolio of blockchain assets upon closing of the portfolio swap smart contract;
receiving, from the blockchain portfolio swap smart contract marketplace, the list of blockchain portfolio swap smart contracts for which a potential buyer has indicated a desire to buy, sorted according to a metric;
transmitting, to the blockchain portfolio swap smart contract marketplace, a desire to sell one of the list of blockchain portfolio swap smart contracts;
receiving, from the blockchain portfolio swap smart contract market, a smart contract address of the one of the list of blockchain portfolio swap smart contracts; and

broadcasting, to a network of blockchain of a funding asset of the one of the list of blockchain portfolio swap smart contracts, a blockchain transaction transferring an amount of the funding blockchain asset sufficient to fund the one of the list of blockchain portfolio swap smart contracts.

18. The method of claim **17**, wherein the operation that transmits the desire to sell one of the list of blockchain portfolio swap smart contracts includes a contingency on the rebalancing parameters of the one of the list of blockchain portfolio swap smart contracts.

19. The method of claim **18**, wherein the one of the list of blockchain portfolio swap smart contracts includes only a portfolio of blockchain assets pegged to an published index of assets, the portfolio of blockchain assets being rebalanced according to changes to the index of assets periodically.

20. The method of claim **17**, wherein the operation that transmits the desire to sell the one of the list of blockchain portfolio swap smart contracts includes a requirement that the one of the blockchain portfolio swap smart contract be rebalanced automatically.

* * * * *