MatterFI^{TM}: Powering A Comprehensive Solution For Blockchain-based Financial Instruments

Chris Odom

and Michal "Mehow" Pospieszalski

chris@matterfi.com

mehow@matterfi.com

May 15th, 2022

Abstract

A comprehensive system, based on open protocols, providing:

- User-controlled identity and payments across all blockchains, based on OBPP-05 payment codes.
- A solution for the problems of receiving addresses in order to have a single, re-usable address across all blockchains, with return addresses, privacy for on-blockchain transactions, and proof that any given deposit, payment or withdrawal really was sent to the identified person.
- User credentials for handling KYC-AML regulatory requirements for regulated parties such as banks, while maintaining users' control over their own identity.
- User-controlled issuance of tokens on-blockchain, for whatever purpose, based on Simple Ledger Protocol.
- Server deposits for off-blockchain transactions of coins and tokens, based on Open-Transactions. [7]
- Currency agnostic integration of multiple blockchains and their derivatives.
- On-blockchain, off-blockchain, and cross-blockchain interactions.
- A solution for theft and fraud wherever funds are deposited with 3rd-party custodians, based on pools of servers that employ multisig storage of funds.

The above feature set allows for hybrid "on" and "off" chain Distributed Finance "DeFi" implementations where custodians can issue cross-chain moving tokens, DeFi smart contracts can run off-chain in specific jurisdictions, on-chain smart contracts can use fast processing off-chain multi-sig endpoints, and touring incomplete coins' smart contracts can be processed off-chain and returned on-chain.

1. Introduction

The Bitcoin blockchain and its derivatives implement a much needed 'foundation layer' of hard money, where transactions are irreversible and censorship-resistant, consisting purely of peer-to-peer payments [6] secured by proof-of-work.

The blockchain ecosystem is also the ideal medium for issuing 3rd-party tokens. These tokens allow the blockchain-based circulation of unique units of value from different issuers. Several protocols already exist, including Simple Ledger Protocol (SLP) [2] for Bitcoin-based blockchains, and ERC-20 for Ethereum.

Blockchain users and custodians need solutions to a variety of identity problems, such as a reusable address that can be used as a public ID and that supports return addresses. Solving the

problems of receiving addresses as we know them today is critical for usability and for unifying identity across all chains. We propose to accomplish this using the OBPP-05 address format in conjunction with identity credential support built into the Open-Transactions library.

Blockchain users also often deposit funds with 3rd-party custodians such as exchanges for the increased functionality available there, like trading on markets. These 3rd-party custodians are historically where most theft and fraud have occurred, and the losses are in the billions.

Unfortunately, servers in use today must be trusted to hold the funds, to accurately maintain their internal ledger, and to faithfully execute the transactions requested by their users. The MatterToken ecosystem solves this problem via the MatterNode which is a server implementing the Open-Transactions protocol. The MatterNode is based on cryptographic proof instead of trust, allowing any willing parties who wish to contract with each other to enjoy the benefits of a server without needing to trust it.

In this paradigm, transaction servers are demoted to mere notaries, only able to countersign contracts that have first been signed by their clients. Since only each client has access to its own private key, receipts are unforgeable by the MatterNode, who can only "timestamp" transactions as they come through, and cannot falsify them.

Such an off-blockchain server, based entirely on cryptographic proofs, is what makes it possible to prevent theft of reserves by storing blockchain coins and tokens in multi-signature voting pools (X-of-Y) that consist of notaries who audit each other. Whenever a withdrawal needs approval, the auditors vote multisig on-blockchain to release funds—but only when authorized by users' signed receipts. No one has so far built such a pool because it is not possible without using a server based on cryptographic proofs. Servers in common use today do not yet offer this advantage.

2. MatterID

Ascertaining and using *identity* is critical for many blockchain-based businesses. What is *"identity,"* from a software perspective? Your social security card or driver's license is not the sum total of your identity, not even in court. Like a marriage license, or a dog license, or even a witness statement in an affidavit; each is just one piece of evidence about you. Or more specifically, each is a claim (or set of claims) about you that has been certified by someone you know, or by some official authority that others trust. These pieces of evidence are like a cloud formed around you.

But even the cloud is not your identity itself. Identity is an ethereal thing. The cloud merely consists of various pieces of *evidence* about your identity, which is the closest we can ever achieve in the real world to tracking a person's identity. This will always be true. Put another way, having the name of "Bill Gates" is not the same thing as being *the* Bill Gates who founded Microsoft. It's not the name alone, nor some singular taxpayer ID number that imbues identity, but rather, those names and numbers are just examples of verified (or verifiable) facts about that person.

A person's true identity can never be tracked as anything more than a set of claims – and claims about those claims – regarding specific facts, relationships and accomplishments in that person's life, that distinguish him from anyone else subjectively, in the eyes of those he is interacting with. The only question that remains is whether the user will create and control his own credentials, rather than an authority.[1]

However, even given a system where users have full control over their own cryptographic credentials, we can still *have* authorities, and authorities can set permissions or revoke access in their own systems. The MatterFi decentralized model of identity provides users with self-determination, while ensuring that various platforms can still regulate their internal systems.

[1] As seen with X.509 certificate authorities.

3. Identifying Users and MatterNodes for AML-KYC or Other Purposes

One requirement for many blockchain-based businesses is to be able to identify a user, and from there to be able to link that user to the signatures they sign, the payments they make, the messages they send, the claims that distinguish their identity, and authentications about those claims made by various authorities. But how can we associate a person's identity to a single ID, given the ethereal nature of identity itself? How can we prove which metadata is *authoritative,* given one person's signature or another's?

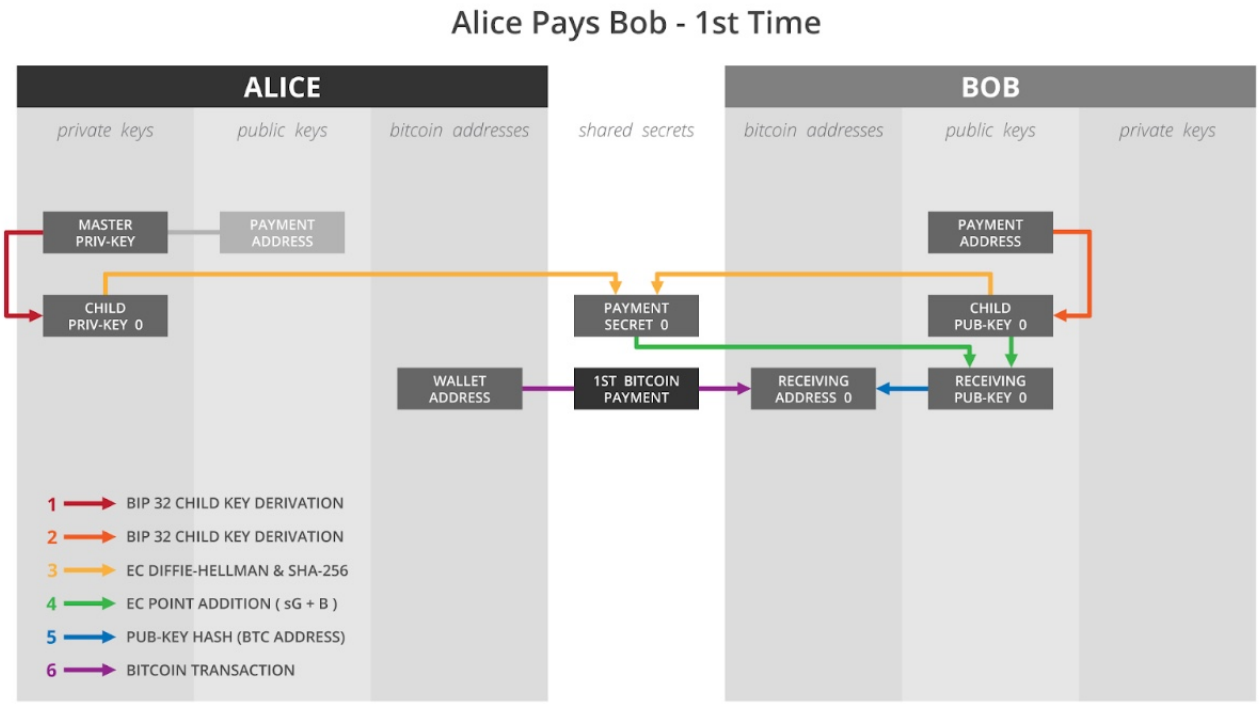
In order to answer this question, we will examine what's possible using the MatterID implementation of OBPP-05 payment addresses, identity credentials, and blind claims.

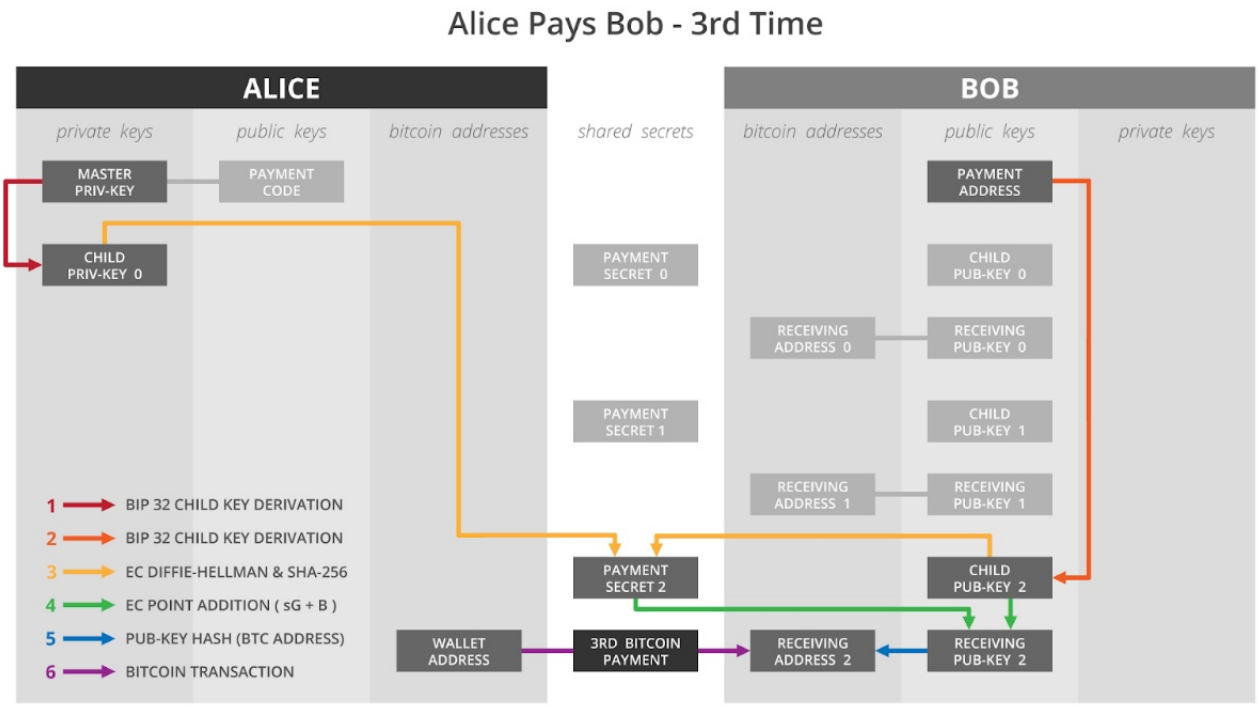
MatterFi employs Open-Transactions identity credentials, which allow users to make claims about themselves regarding their facts, relationships, and accomplishments. Moreover, anyone else, including trusted authorities, as well as colleagues, family members, etc. can sign verifications of those claims. And it is in those authenticated claims where we can find the proofs needed for identifying each person according to the needs of each scenario, such as when an exchange performs a withdrawal for one of its users.

But first, we must be able to link a set of credentials to a specific re-usable ID and its signatures. And for that, we start with OBPP-05.

4. OBPP-05

The *OBPP-05 Payment Code* is a way to format blockchain addresses so that each user can have a single, re-usable address across all blockchains, yet the address itself does not actually appear in any on-blockchain transactions, and thus cannot be observed publicly on-blockchain by any third parties, unless they have the party's private key. Instead, the receiving addresses are calculated deterministically via *"spooky action at a distance"* using a Diffie-Hellman shared secret. [15] This is a process whereby each party, using his own private key and the other party's public key, is able to calculate a shared secret key, which no one else can calculate without one of the private keys. The parties also increment an index after each transfer, so there is a new blockchain receiving address calculated for each transaction between them *(see diagrams).*





The calculation, transfer, storage, and maintenance of new Bitcoin receiving addresses has up until this point been a huge headache for Bitcoin merchants, exchanges, and users, who usually need to generate and track a new receiving address for each and every transaction. Receiving addresses really are a low-level detail that should not require user involvement, and OBPP-05 makes that experience a reality for the user, while restoring some privacy and fungibility to on-blockchain transactions, and while simultaneously allowing users to know *for certain* who they have paid and who is paying them, without having to constantly exchange new receiving addresses with one another.
Justus Ranvier, chief architect at MatterFi (and the inventor of OBPP-05) said:

> *Payment codes are a technique for creating permanent Bitcoin addresses that can be reused and publicly associated with a real-life identity without creating a loss of financial privacy. They are similar to stealth addresses, but involve a different set of trade-offs and features that may make them more practical. [9]*

It's important to note that payment codes do not require any changes in the Bitcoin protocol itself, and so adoption has already begun. Bitcoin wallets including Billion, Samourai and MatterWallet have already adopted the technology.

The OBPP-05 payment code provides users with a single ID that can be used across all blockchains, without having to generate a different ID for each blockchain. The same ID will work on Litecoin, Bitcoin, Bitcoin Cash, etc.—any blockchain where the user holds a private key.

I. Useful Properties of OBPP-05 in Distributed Finance:

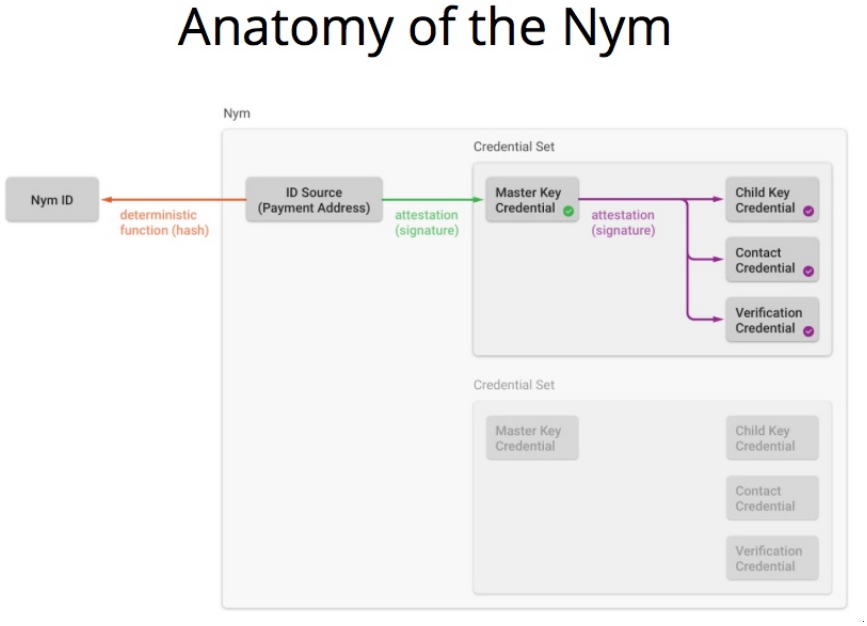
- A user's payment code can be freely published everywhere and serve as his 'public ID' without fear of losing any security or anonymity.
- Users only need to exchange payment codes *once.* Thereafter, messages and payments, agreements, and smart contracts, including DiFi contracts themselves can safely flow between them, without having to exchange a new address each time.
- The same payment code works across *all Bitcoin-derived blockchains*, such as Litecoin, BSV, and Bitcoin Cash, as well as on-and-off blockchain. MatterFi's roadmap also includes support for other popular chains like Ethereum and EOS.
- The same OBPP-05 code that is used for payments can also be used for messaging, and for signing messages that are sent, as well as for verifying signatures on messages received.
- A third party public observer will *not* see your payment code in any of your actual on-blockchain transactions. This is because the on-blockchain transactions use receiving addresses that are generated deterministically by both parties without having to transmit anything, using a Diffie-Hellman shared secret *("spooky action at a distance.")* [13].
- The parties to a transaction can see their entire history between each other. This is possible because each party has one of the private keys needed to generate the deterministic receiving addresses used by the other party.
- This means that when a message or payment is sent, a sender knows for certain who the recipient is, and there is no confusion that the recipient has provided some third party's receiving address instead of their own.
- This also means that when a user receives a payment, the recipient knows for certain who the sender is. The recipient will see the sender's *"display name"* on the incoming payment, and can even hit 'reply' to send a payment or message back to them. Bitcoin ATM operators constantly ask about this functionality.

The above identity features of the MatterFi ecosystem allow for easy to use world wide on- and off-chain DeFi. For example, a user can enter into a smart DeFi contract (which has its own cryptographically signed OBPP-05 identity) and the smart contract can receive and send funds to the user having a mathematical guarantee that the funds are coming and going to the exact same OBPP-05 identity. Meanwhile the smart contract (whether its running on some chain or off-chain on a MatterNode) can maintain a mathematically provable audit trail of it own transactions while still co-existing in a de-centralized ecosystem. This allows for interesting use cases such as a on-chain DeFi smart contract that executes a portion of its code in a certain physical location. In such a case, the MatterNode operator would simply have a 3rd party digitally sign their Nodes OBPP-05 identity to certify that the server is in fact in a certain location. Government financial regulators would be ideal location certification identities.

5. MatterID Credentials

In this section we discuss how we implement OBPP-05 such that additional data (such as certifiing a MatterNodes location) is implemented in our identity credential scheme with OBPP-05. It's not enough just to prove that a signature really was signed by the expected OBPP-05 ID. We must also be able to attach **metadata** to that ID in the form of *credentials*, consisting of *claims* and *claim verifications* by various authorities.

In crypto software, a *pseudonyum* (or simply *"Nym")* is a potential identity who has a public address (his OBPP-05 ID in our case), a corresponding private key, and a set of credentials associated with them. However, that doesn't mean yet that any real-world information has been authenticated regarding the *actual person* behind that Nym.



$$PaymentAddress + Credentials = Nym(pseudonym) \tag{1}$$

Bitcoin wallets often disappointingly show *your own receiving address* (not the sender's address) when displaying a record of received funds. But with OBPP-05, when you receive an on-blockchain payment, there *is* a display name for the sender. This is because the Opentxs API has already downloaded the sender's *identity credentials*[2], which are cryptographically verified against his OBPP-05 address. The *display name* field is one good example of the sort of metadata that users can attach to their credentials via signed claims.

A user's credentials contain other claims made about himself, regarding various facts, relationships, and accomplishments. For example, *"My SSN is XXX-XX-XXXX, signed Bob."* His claims may also be *verified* (signed) by other users, and those *claim verifications* may also appear on his credentials. [14] For example, a verified claim might look like this, if we imagined it in human-readable form: *"The hash fingerprint of an encrypted JPG of my scanned driver's license is 1Jasd876y2jhg34h998, signed Bob"* followed by: *"We have officially authenticated this information in*

[2]See ContactManager::NewContact(), https://github.com/Open-Transactions/opentxs/blob/develop/include/opentxs/api/ContactManager.hpp

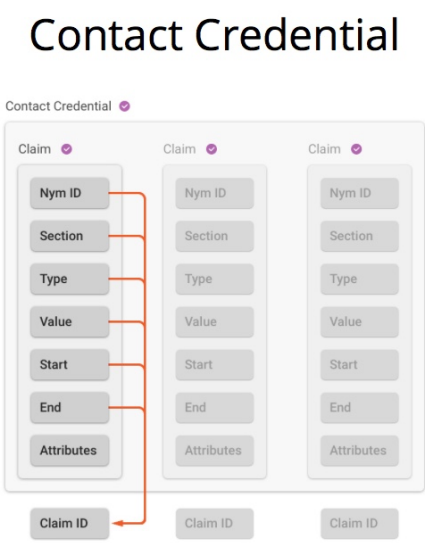


Figure 1: *Contact Credential*

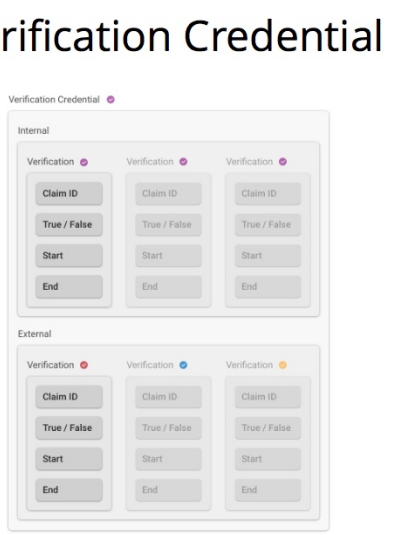


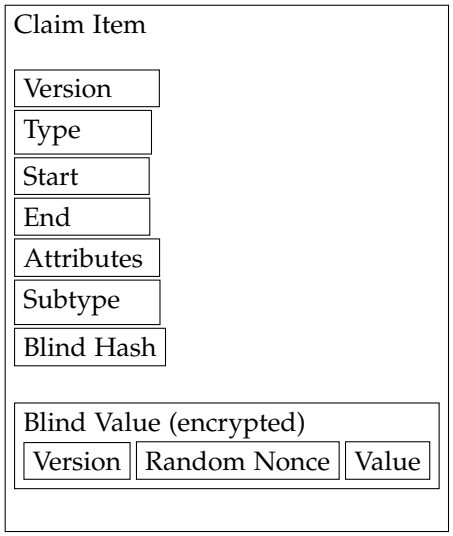
Figure 2: *Verification Credential*

person and can reproduce it upon demand, ***signed,*** *Some Trusted Authority."* Note that digital signatures are unforgeable.

For example, when you first meet someone and add them as a new contact by scanning their QR code, MatterWallet creates *reciprocal verified claims* between both parties, proving to themselves later that they have DeFinitely met each other in real life. This way, a *"web of trust"* can form organically, without any need for *"key signing parties"* as envisioned in the earlier days of crypto. Various trusted authorities will be responsible to identify users and verify claims, as we shall explore below. That way, users will be able to prove their identity to one another, without having to ask the authority, since his signature already appears on the claim verification.

6. Blind Claims

User credentials will also take advantage of *blind claims,* meaning those claims are proven in a 'blind' way. To do this, the *value* field of the claim is encrypted, and the Trusted Authority signs a hash of this *blind value* instead of signing the plaintext value as might normally otherwise occur. However, they have still seen and verified the plaintext value, before signing the claim verification. This blinding is often desirable. For example, there may be legal requirements in certain situations to prove a person's age, and so it would be useful for users to be able to prove their age and photo when necessary, *without* having to post their birthdate or photo publicly for the whole world to see. This is possible using a blind claim.
In Open-Transactions the structure of a blind claim is the same as for a normal claim, except there is no *value* field. Instead, the claim includes a ***blind value*** field that's encrypted to a (deterministic) key; structured like this in Open-Transactions:



Note that the blind value itself is encrypted to a deterministic key which is recoverable from the user's seed.[3] Publicly only the blind hash is visible on the user's credentials, as well as any signed verifications of that claim.

For example:

- The State of California could sign the hash of your encrypted birthdate in order to allow you to prove your age on demand, when required.
- Alternately, someday you may be able to properly identify yourself to a notary public, and then he'll take your picture on the spot, and sign the hash of an encrypted JPG of your face. So when you are signing documents, not only is your signature provably the correct signature from your OBPP-05 ID, but you can also reveal your photo to any challenger while simultaneously proving that an *authority* has certified that they created that photo, and authenticated it for *that same OBPP-05 ID.*
- A notary public – or an on-boarding department – could additionally scan your driver's license and then **sign the hash of an encrypted JPG of your driver's license.**

[3] Otherwise we might be forced to use a *deterministic nonce* instead of a *random nonce.*

- The possibilities are endless. *Any* data can be authenticated and thus made available for users to prove to others in the future, without having to reveal that data publicly, and without having to re-verify their identity over and over again to an on-boarding service over a webcam.

- We must consider that any data whatsoever that is of value when it's been authenticated, ***will* therefore end up being authenticated** by someone or another. Organizations will spring up to authenticate the data and sign the appropriate credentials.
  - Sherpas will certify that claimants have actually climbed Mount Everest.
  - The Trusted Authority will certify claimants with *Approved* or *U.S. Person* status.
  - Colleagues will certify they have worked with each other in the past.

Other forms of proofs, including zero-knowledge proofs, will eventually be added to user credentials. For example, zk-SNARKS and zk-STARKS are promising innovations that will likely be used for various blind or transparent proofs.

7. Open-Transactions Protocol

The Open-Transactions (OT) project is a collaborative effort to develop a robust, commercial-grade, fully-featured, free-software toolkit that implements **off-blockchain** transactions purely as **cryptographic proofs.** The MatterFi ecosystem consists of proprietary and open source software which implements the open source OT protocol.

I. Financial Cryptography

Open-Transactions uses strong cryptographic techniques to create secure financial instruments, such as digital signing to create non-repudiable contracts, and homomorphic encryption to create untraceable digital cash. In OT, transactions are unforgeable, receipts are destructible, and balances cannot be falsified or changed without user consent. OT is able to prove all balances, as well as which instruments are valid, and which transactions have closed, without storing a receipt history, as long as you have the last signed receipt.

Open-Transactions implements financial instruments as *Ricardian Contracts*, which are contracts that can be understood by humans as well as manipulated by software [3].

All contracts in OT use the same basic structure: all parties involved sign an agreement which is notarized by an independent third party witness. This technique is known as Triple-Signed Receipts. Importantly, transactions are formed and signed on the client side first, before being notarized by any server. An OT client is thus ensured that an OT MatterNode server cannot falsify his receipts against his will, since the server can't forge the client's signature.

This basic structure can be built upon to create many types of financial instruments.

II. Financial Instruments

We DeFine an off-blockchain transaction as a group of operations on contracts capable of objectively proving balances (and changes of balance) between adversarial parties.

All transactions use the same basic structure: the parties involved sign agreements which are then countersigned by an independent MatterNode server. Transactions are irreversible since the receipts are always formed and signed on the client side first, before being notarized by any server. This prevents the MatterNode from falsifying receipts, since it can't forge the client's signature.

This basic structure can be built upon to create many types of financial instruments. Those supported by Open-Transactions include:

- **Transfer.** An atomic movement of funds from one account to a different account, like a bank account-to-account transfer.
- **Cash.** Untraceable cryptographic tokens which can be securely redeemed by the recipient without revealing the sender.[4]
- **Cheque.** A payment which is not deducted from the sender's account until the recipient claims it.
- **Voucher.** A payment which is deducted from the sender's account at the time of creation.
- **Invoice.** A payment request which the recipient can opt to pay from any of his accounts.
- **Market Offer.** An open agreement to exchange a given quantity of one unit type for a given quantity of another unit type.
- **Recurring Payments.** An agreement between two parties that includes an optional initial payment, followed by a set number of additional payments over a specified period of time.

[4]Based on work by David Chaum [1]. Open-Transactions currently includes Lucre [5] by Ben Laurie.

- **Smart Contract.** A customizable agreement between multiple parties, containing user-DeFined scripted clauses, hooks, and variables. This is exactly how DeFi is currently implemented except that thus far people have only implemented them on chain instead of both on- and off- chain.

We note that OT natively supports DeFi smart contracts which makes it possible with our system to execute DeFi off-chain where the Node operators set the transaction fees (or make them free). This allows for new financial implementations that require real time processing and operator set-able transaction fees. One example would be quant trading. Furthermore, the entire smart contract need not be off-chain. With our system an infinite realm of hybrid on-and off-chain DeFi is possible.

III. Destructible Receipts

In the solution we propose, each MatterNode-signed receipt contains a copy of the original user-signed request.[5] Since that request also includes a statement of the balance, we can always prove the current balance using the most recent receipt.[6]

In addition, we can prove which instruments are still valid by including a list of open transaction numbers on each signed request [8]. The request also includes a list of any incoming transactions; each transaction must use an open transaction number.

At all times throughout the process, all parties using Open-Transactions are able to prove their current balance, as well as which instruments are valid and which transaction numbers are closed, without storing any history except their last signed receipt. Alice's own signature proves these things to Alice, and Bob's own signature proves these things to Bob. The MatterNode is unable to defraud them.

Does this mean that parties *should* destroy their historical receipts? Not necessarily. But it should be noted that parties are not *forced* to keep their receipt history in order to prove the current state. Proofs which require a full history are $O(n)$, whereas Open-Transactions proofs are $O(1)$. $O(1)$ balance proofs are preferable to $O(n)$ proofs, because even though most users would choose to save their transaction history, the risk of a balance proof failing due to data loss does not grow without bound over time.

IV. Counterfeiting

Since a MatterNode is unable to falsify Alice's receipts against her will, the only crime left is counterfeiting funds with a dummy account. That is, even without falsifying any of Alice's receipts, a MatterNode can still create a dummy account and then sign a false receipt for that dummy account, and thus create counterfeited funds which can then be spent to Alice or Bob.

Fortunately, counterfeiting can be easily prevented by auditing the receipts. But while it is technically easy to construct an audit server, we still need a party who is incentivized to operate that audit server.

V. Voting Pools

The most ideal solution is for users to deposit their on-blockchain funds (coins or tokens) into a multi-signature voting pool composed of several notaries who all audit each other, voting on-blockchain when necessary to approve withdrawals. This is only possible if the off-blockchain

[5] Based on work by Ian Grigg [4].
[6] Based on work by Bill St. Clair [18].

system is based on cryptographic proofs, as Open-Transactions is.

This Voting Pool arrangement solves two problems at once:

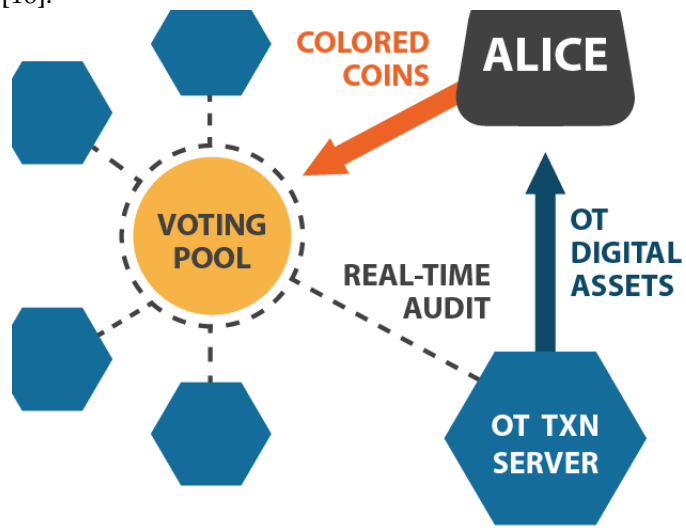
1. Counterfeiting. Multiple parties are auditing each MatterNode, which prevents it from counterfeiting on its internal ledger.
2. Theft. An individual MatterNode is also incapable of stealing coins out of the pool, since a multi-signature vote is necessary to retrieve the funds. (And those votes are controlled by the auditors.)

8. MatterFi Voting Pools

Currency exchanges and other trading platforms usually desire to perform order matching more rapidly than what is possible on the blockchain itself. These services accept custody of user funds, perform transactions in a separate off-chain system, and use a database to track customer balances. Typically these services are not cryptographically secured, or independently auditable. Customers also give full control of their deposited funds to the custodial service, which exposes them to the risk of theft or loss of their coins.

Unlike legacy currencies, cryptocurrencies can be irrevocably lost or stolen, and it's typically not possible to distinguish between insider or external theft. Historically, this ambiguity appears to have been routinely exploited.

We propose *voting pools* as an open standard intended to be a universal replacement for bespoke systems that handle customer cryptocurrency deposits. A voting pool is an arrangement of notaries to securely store and account for customer cryptocurrency deposits, and to redeem valid withdrawal requests even in the event the customer's MatterNode of choice has completely disappeared. Voting pools are designed to ensure that no single person or organization can ever perform unilateral actions on deposited funds, in order to reduce the risk of loss or theft, and custodial liability [12]. Each MatterNode in the voting pool operates its own audit server, and each auditor has a corresponding blockchain wallet. The blockchain wallet manages the multi-signature transaction generation, as well as a hierarchical and deterministic list of addresses for bitcoins [17] and colored coins [16].



When a customer deposits cryptocurrency into a voting pool, he receives corresponding units in his account on his MatterNode of choice [10]. How? Each audit server watches the receipt stream for requests to deposit or withdraw bitcoins or colored coins from the voting pool, and then communicates with its Bitcoin wallet as appropriate. The auditor independently verifies the Open-Transactions operations of all notaries in the voting pool, as well as the bitcoins held by the pool on the blockchain itself. The auditor uses this audit data to know when it should direct the wallet to create a withdrawal transaction [11], and it is also the component responsible for information sharing and achieving consensus between all members of the pool. Effectively each auditor conducts a permanent, real time proof-of-reserves audit against all of the notaries in the pool, and simultaneously enforces it. It is the auditors and the wallets who hold the keys

to creating blockchain transactions at the request of the user, and the auditors must all act by consensus and with the cooperation of the wallet to create multi-signature blockchain transactions.

Each Voting Pool regardless of how many servers aka notaries it contains is still implemented as a single MatterNode with its own OBPP-05 identity that supports on-chain mulitsig.

I. Security Goals

In order to achieve the desired security and robustness goals for voting pools, the following criteria are enforced:

- Customers should be strongly discouraged from reusing deposit addresses. The voting pool itself must never intentionally reuse a bitcoin address.
- All Bitcoin addresses used by the pool must be deterministic for auditing purposes. Each member of the pool should be able to calculate all members' series of deposit and change addresses.
- Withdrawal transaction input selection must be deterministic in order to minimise the cost of coordinating transaction signing.
- It must be possible to keep a majority of the private keys offline for security reasons, and bring them online as needed to process withdrawals.
- It must be possible to alter the voting pool by adding, removing, or replacing members in a coordinated and secure fashion. It is necessary to use *Smart Properties* to accomplish this. These are described in the section below on colored coins.

II. Security Model

The goal of the voting pool security model is that users of deposit-accepting services should never experience a loss of deposited funds.

We can group the various ways in which this goal might not be met into three general categories:

- Type 1 Event (Theft/Loss). A user permanently loses their funds because a third party has gained control of them without the user's consent, or because the private keys needed to spend them have been irrevocably lost.
- Type 2 Event (Denial of Service). A user temporarily loses some or all of their ability to use their funds, but no third party has gained control over them.
- Type 0 Event. Type 0 Events will be used to describe all other abnormal conditions from which the pool must recover which do not directly involve a loss of customer deposits.

III. Security Theorem

If the probability of $m + 1$ (Type 1 Event) or $n - m + 1$ (Type 2 Event) services simultaneously and identically behaving in a malicious or incompetent manner is lower than the probability of any individual server behaving in a malicious or incompetent manner, user deposits on that service are at less risk of loss if the service is a member of an $m - of - n$ voting pool than they would be at risk if the service is not a member of a voting pool.

Voting pools can guarantee the integrity of user deposits if, in any given situation, at least m pool members are well-behaving for Type 1 events and at least $n - m$ pool members are well-behaving for Type 2 events.

IV. The MatterAuditor

The MatterAuditor listens to the Audit Streams broadcast by all the Notaries and independently verifies them for correctness. The same stream which carries regular OT transaction information also contains the OT receipts for Bitcoin withdrawal requests from the pool. The auditor initiates or authorizes a blockchain withdrawal transaction via the wallet if and only if the audit for that service is clean (verified correct).

The auditor is responsible for maintaining an independent copy of the same deposit database as the transaction server. It also tracks withdrawals from the time at which it receives an OT receipt, containing a withdrawal request, until the corresponding Bitcoin transaction is fully confirmed on the blockchain.

All messages which must travel between the transaction server and the blockchain wallet pass through the auditor.

In order to create withdrawal transactions, wallets must be able to select inputs and change outputs, and calculate the minimum required transaction fee deterministically. In order to achieve determinism, the sequence of withdrawals must be globally consistent. Before sending any withdrawal request to the wallet, the auditors are responsible for achieving consensus on a serialization order for withdrawals.

The MatterAuditor is proprietary software which is operated by staking MatterTokens.

V. MatterNode

The MatterNode must keep track of all blockchain-denominated balances via OT receipts. In addition to the separate account(s) for each customer, the server must track a service account to hold the blockchain-denominated balance owned by the service itself.

If necessary, the server should also maintain an application account to hold the balance of any funds which are being manipulated by an external system.

For example, in the case of a high-frequency exchange, the application account would belong to the order matching engine. When a customer enters a trade, the exchange front-end will call the applicable OT functions to transfer the appropriate balance from the selling customer's OT account to the application OT account, and also from the application account back to the the appropriate purchasing customer's OT account. Any trade fees that the exchange earns would be sent to the service account as part of the transaction. This separation of application account, service account, and customer account, prevents the mingling of funds.

The MatterNode is also responsible for passing PaymentRequests from the Auditor to the customer, crediting customer balances after the successful receipt of a blockchain deposit, and passing withdrawal requests to the rest of the voting pool via the audit servers.

The MatterNode must maintain a permanent deposit database containing each PaymentRequest generated for that server and its associated status (number of blockchain confirmations and the OT receipt crediting the appropriate Nym with the deposit)

VI. Audit Stream

The MatterNode must broadcast five types of messages in an indexed and hash-chained sequence which form an Audit Stream.

The five types of messages are:

- Update to an inbox
- Update to an outbox
- Update to an account balance file

- Update to a Nym box file
- All MatterNode replies to transaction requests.

VII. OT Client

In order for users to deposit cryptocurrency into the voting pool, the OT Client must be capable of parsing, verifying, and if necessary forwarding to another blockchain wallet client, payment requests. This is necessary to ensure a malicious server can not send a fake deposit request that results in a customer sending funds to an address not controlled by the pool.

All users of services which are part of a voting pool must have an OT client running on their device in order to use the service. The service front end can communicate transparently with the OT Client via a local websocket interface.

The OT client will be capable of operating in the background with the bare minimum necessary interaction in order to provide the lowest possible disturbance of the front end service's user experience (UX).

VIII. Voting Pool Key Management

Each server in the pool requires an offline, air-gapped machine for key generation called a key server. It is equipped with either a dedicated, non-networked printer, or else a CDR drive. No media of any kind is ever allowed to cross the air gap in the online->offline direction.

The key server generates random BIP32 seeds in batches (default: 52, or enough for one year). When a batch is created, it prints the xpubs (extended public keys) for all 52 seeds on paper as QR codes (alternately on a virgin CDR which is discarded after a single use). This paper is then manually walked to the auditing server and scanned. The auditing server adds each xpub to the keyset DeFinition.

At the same time, the key server also prints two redundant copies of the QR codes containing the xprivs (extended private keys), one per page (one per CD) which the service should hold in some physically secure fashion and back up securely. It is not necessary for all individual services to take extraordinary measures to protect the private keys from physical destruction, since the pool can tolerate a loss of keys that involves less than (n-m) members. One copy held in an offsite location with another copy held on site is sufficient.

Xprivs are loaded into the auditing server in series number order to create the hot series. Each participant in the pool should have a method of being notified when the hot series is close to being exhausted so that an employee can be instructed to load the next xpriv into the auditing server.

New key batches should be generated early before the old batch is consumed (default: 75 percent used). If for any reason one of the participants is late and does not generate a new batch on time, the last DeFined series number is used for accepting deposits until the administrators of the other pool members can correct the situation.

The key server must also be equipped with a scanner. Prior to putting any keys into service, they must be validated.

Validation procedure: The key server will create the first one million public and private keypairs from each seed in the batch, sign a nonce with each private key and verify the signature with the corresponding public key.

Then the user will scan in the printed public and private keys, and the key server will verify the scanned versions match the original versions and repeat the million keypair test.

Both versions of the test must match identically before placing any of the keys into service.

IX. Standard Pool Sizes

Every pool represents a compromise between performance and cost. For security and reliability purposes, higher reliability levels are better, however they must be balanced against the cost factor. Standard pool sizes are the lowest cost pools that produce a given reliability level.

Reliability Level	Theft Resistance Factor	Redundancy Factor	Cost Factor	Pool Size
1	2	1	5	2-of-3
2	3	2	8	3-of-5
3	4	3	11	4-of-7
4	5	4	14	5-of-9
5	6	5	17	6-of-11
6	7	6	20	7-of-13
7	8	7	23	8-of-15

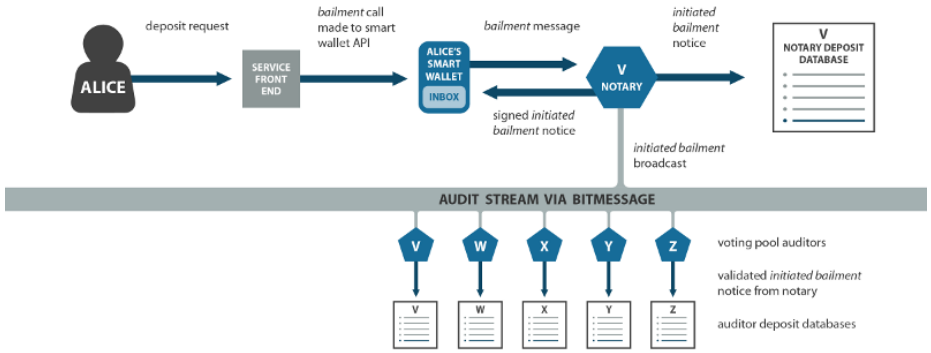
9. Voting Pool Deposit Process

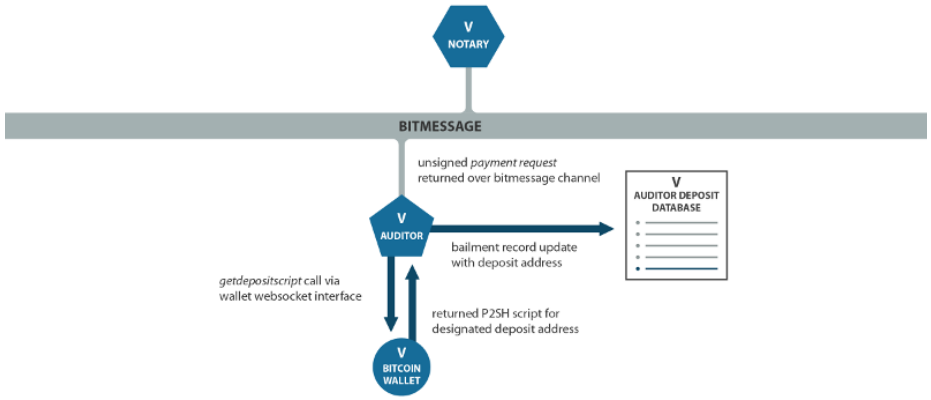
I. Initiation

Customers will normally request a deposit address by interacting with the service front end web site or some other software application. When the service receives such a request, it notifies the OT client via the OT client API function: *requestBailment*.

When the OT client receives notice of a user desire to deposit funds to a voting pool, via any method, it sends a *bailment* transaction request to the MatterNode to initiate the deposit process.

The MatterNode validates this request, and replies with a signed receipt. A copy of this receipt is broadcasted to the *audit stream*, and another copy is stored inside an *initiatedBailment* notice that's placed in the user's inbox. The MatterNode adds this association to a *bailment database* for future reference.





II. Payment Script Generation

When an audit server validates the MatterNode's reply to the bailment message from the MatterNode to which it is assigned, it adds the receipt identifier to its *bailment database* and calls *getDepositScript* via the websocket interface to the blockchain wallet, using the address identifier for the next unused deposit address.

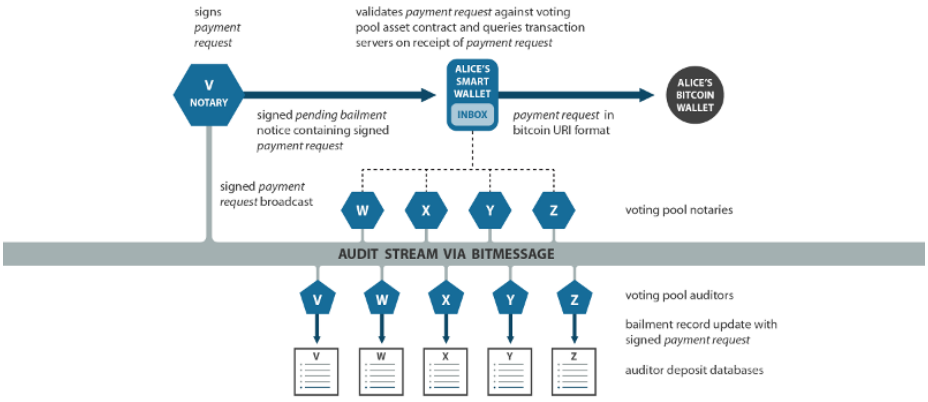
The wallet calculates and returns the designated deposit address as P2SH output script. The audit server uses this information to update the bailment database, and to assemble and sign a *PaymentRequest*.

III. PaymentRequest delivery

The audit server broadcasts the *PaymentRequest* to all notaries and auditors in the pool. The MatterNode replaces the user's *initiatedBailment* notice in the inbox with a *pendingBailment* notice containing a copy of the *PaymentRequest*.

When the other audit servers in the pool receive the *PaymentRequest* broadcast they add the deposit to their respective bailment databases. The other notaries in the pool cache the *PaymentRequest* to answer verification requests from the OT client.

The OT client should validate the *PaymentRequest* against the voting pool asset contract. If it is valid then it should query a random selection of other notaries, at least $m - 1$, using the *PaymentRequest identifier* to verify whether they have seen it. If this check is successful, it then initiates the blockchain transaction by passing the *PaymentRequest* to the user's local wallet application which is configured to handle [[*bitcoin* :]] *URIs*.

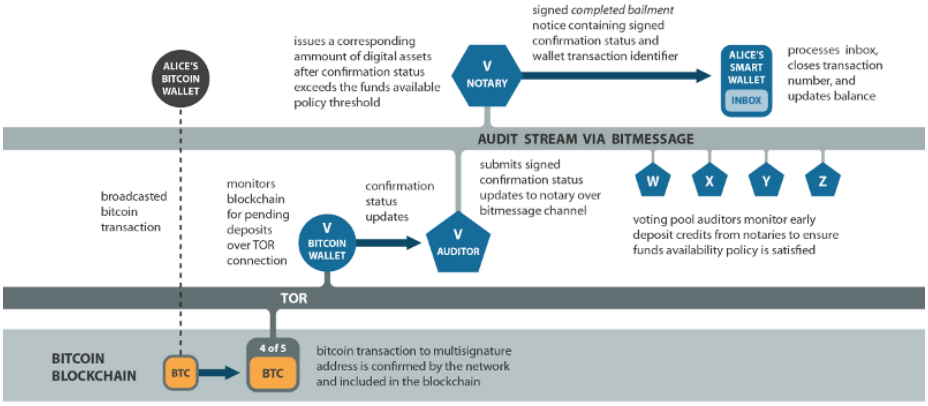


IV. Deposit and Crediting

The audit server broadcasts the *PaymentRequest* to all notaries and auditors in the pool. The MatterNode replaces the user's *initiatedBailment* notice in the inbox with a *pendingBailment* notice containing a copy of the *PaymentRequest*.

When the other audit servers in the pool receive the *PaymentRequest* broadcast they add the deposit to their respective bailment databases. The other notaries in the pool cache the *PaymentRequest* to answer verification requests from the OT client.

The OT client should validate the *PaymentRequest* against the voting pool asset contract. If it is valid then it should query a random selection of other notaries, at least $m - 1$, using the *PaymentRequest identifier* to verify whether they have seen it. If this check is successful, it then initiates the blockchain transaction by passing the *PaymentRequest* to the user's local wallet application which is configured to handle [[*bitcoin* :]] *URIs*.



Type 1 Event: Fraudulent Deposit Address. A malicious or hacked operator may give the customer an invalid PaymentRequest in an attempt to steal deposits.

Each pool member's Bitcoin wallet must notify its audit server of any deposits received to an address which the pool controls. When an incoming transaction is received to an address the audit servers are expecting due to previously broadcast *PaymentRequest*, they will use the *getinfomultisigwalletaddress* calls to identify the incoming transaction, and *gettransactionstatus* to monitor its confirmation status.

Type 0 Event: Deposit Never Received. It's possible that the customer may never actually transfer bitcoins after requesting a PaymentRequest.

Type 1 Event: Unknown Deposit. A deposit may be received to an address which has never been used, and for which a PaymentRequest was never created so no member of the pool knows to which nym it should be credited.

Type 1 Event: Duplicate Deposit. A deposit may be received from an address which has been previously used, so the audit servers know to which nym the address is assigned.

Type 0 Event: Dust Handling. The size of the deposit may be below the network dust threshold (small enough that it would require more in transaction fees to spend than it is worth).

The audit server relays the number of confirmations the incoming transaction has received to the MatterNode. Once the number is sufficient according to the Funds Available Policy the MatterNode will issue an OT asset for the amount of the deposit to the nym of the user.

The MatterNode will replace the *pendingBailment* notice (in the inbox) with a *completedBailment* notice, which includes a signed copy of the original bailment request, as well as a copy of the audit server's signed notice of confirmations, which includes the transaction identifier provided by the blockchain wallet.

The OT client processes the user's OT account inbox, removing the *completedBailment* notice, which simultaneously closes the transaction number and updates his OT balance.

The audit servers in the pool must monitor all deposits to ensure the Funds Available Policy is satisfied to avoid the risk of a double spend or chain fork causing insolvency. Any server which

offers more early deposit credits than what it can cover with its service account must have its audit status set to degraded.

Type 2 Event: Non-credited Deposit. The MatterNode fails to place a *completedBailment* notice in the user's inbox after a successful Bitcoin transfer.

If an initially-seen deposit fails due to a chain fork, and if the user has not yet been credited with an OT receipt for the deposit, the status of the deposit remains pending. The audit server should notify the MatterNode by setting the number of confirmations back to zero. In the typical case of blockchain reorg event, the deposit transaction should re-enter the mempool automatically and the wallet can assist with this by rebroadcasting it.

If an initially-seen deposit has become invalid due to conflicting transactions which made it into the blockchain, the audit server should mark the deposit as failed by setting the number of confirmations to -1. The audit server should notify the MatterNode of the failure, who then replaces the user's *pendingBailment* notice with a failedBailment notice. The MatterNode should update the status to failed in the bailment database and the address should not be intentionally reused. The OT client can then process the user's inbox, removing the *failedBailment* notice and closing the transaction number. In this case there is no change to the OT account balance, unlike with a *completeBailment* notice.

Type 1 Event: Reversed Deposit. A deposit could disappear from the blockchain after the user has already been issued an OT receipt

10. Voting Pool Withdrawal Process

I. Initiation

Customers will normally request a withdrawal of bitcoins from the pool by interacting with the service front end web site or some other software application. When the service receives such a request, it notifies the OT client via the OT client API function: outBailment.

When the OT client receives notice of a user desire to withdraw funds from a voting pool, via any method, it sends an outBailment transaction request to the MatterNode containing the destination Bitcoin address where the withdrawal should be sent, the amount of the withdrawal, and an extraFee value. The extraFee is added to the transaction fee required by the Bitcoin network and is paid directly from the user's balance and may be zero.

Note: Some customers may wish to have additional restrictions placed on withdrawals, for example to prevent withdrawals to arbitrary Bitcoin addresses, or to require two-factor confirmation of withdrawals, or time delays to allow for notification and manual review of withdrawal requests. All this functionality and more can be provided by users electing to store their deposits in an OT smart contract instead of a standard receipt. Discussion of the capabilities of OT smart contracts is outside the scope of this document.

After the MatterNode receives the outBailment transaction request, it removes the total amount from the user's balance and places it in the outbox as a pendingBailout receipt.

Type 2 Event: Withdrawal blocking. The MatterNode handling a customer account may fail to respond to a valid withdrawal request.

When the auditors see the pendingBailout receipt, they create an entry in their withdrawal database and add the pendingBailout to their queue for the next consensus.

II. Consensus Round Transaction Formation

Each time a new consensus is finalized, the auditors begin processing the specified pendingBailouts (if any) by passing the address identifier of the first input to be used, the address identifier of the first change address to be used, and the withdrawal identifier of each output to the Bitcoin wallet via the *startwithdrawal* API call.

The *startwithdrawal* API call accepts a list of outBailments to process, and the set of parameters which are needed to ensure the transaction process is deterministic.

When the wallet receives this call, it processes the list and parameters per the transaction construction algorithm and returns a list of signatures and status information for each *outBailment*.

If the wallet requires the private keys from additional series in order to fulfill the outBailments, that information will be returned with the status information.

III. Consensus Round Signature Exchange

The auditor takes the signature list from the wallet and broadcasts it to the rest of the auditors. It also collects signature lists from the other auditors and queues them for delivery to the wallet.

IV. Consensus Round Transaction Fee Accounting

Before the auditor can provide the missing signatures to the wallet, it must ensure the transaction fee has been properly accounted for. While the auditor is broadcasting and gathering signatures, it also sends a *txFeeNoticemessage* to each MatterNode from which a withdrawal has been processed, indicating their share of the blockchain transaction fees included in that consensus round.

Blockchain transaction fees are allocated to a specific server by the fraction of total withdrawals in the round which originated from that server. The method of calculating the individual shares should ensure the individual totals add up exactly to the blockchain transaction fees consumed by the transaction.

When the MatterNode receives at least m identical txFeeNotice messages, it then performs a balance adjustment by subtracting the amount from its service account for the pool, and adding that amount to the issuer account for the pool. The auditor cannot release the rest of the signatures to the blockchain wallet until it validates the appropriate balanceAdjustment notice in the audit stream.

Type 0 Event: Transaction Fees Accounting. The originating MatterNode may fail to deduct blockchain transaction fees from its service account and broadcast this receipt in the audit stream.

When the auditor validates all needed balanceAdjustment notices, it delivers the signature lists to the wallet via the updatewithdrawal API call. The blockchain wallet then adds signatures to the transaction(s) until it has m, then it broadcasts the transaction(s) to the network. It is not necessary for each wallet in the pool to include the same list of signatures in the transactions they broadcast. As long as all the transactions are valid, the network should accept one version and include it into a block.

Type 2 Event: Transaction Signature Error. All or a portion of the signatures a wallet receives from the other pool members may be invalid for the given transaction.

The startwithdrawal API call returns a list of one or more normalized transaction identifiers (ntxid), where each ntxid is linked to a list of the withdrawal identifiers corresponding to the outputs in the transaction. The auditor updates its withdrawal database with the ntxids, and begins tracking the confirmation status of the withdrawal. It reports all this data to the MatterNode, and each new confirmation of the transaction.

V. Completion

Like deposits, withdrawal transactions are not considered final until *maturationTime* confirmations have occured. Once the auditor reports a number of confirmations greater than or equal to *maturationTime* for the blockchain transaction associated with a pendingBailout receipt, the MatterNode replaces that receipt with a *completedBailout* receipt. When the auditors see the *completedBailout* receipt in the audit stream they can prune the associated entry from their withdrawal database and stop sending confirmation updates.

Type 2 Event: Failed Withdrawal Transaction. The Bitcoin network may fail to confirm any version of the withdrawal transaction.

11. Token Issuance

Various techniques make it possible to issue new units onto an existing blockchain. For example, ERC-20 on the Ethereum blockchain, and Simple Ledger Protocol on the Bitcoin Cash blockchain. These tokens can be stored and transacted using wallet software, as well as deposited onto servers for off-blockchain transactions such as exchange. The following is a brief introduction to the concepts and techniques involved.

The MatterFi ecosystem allows anyone worldwide to download our Node software and operate distributed finance applications including issuance and processing of their own token or existing tokens.

I. Colored Coins

The term "colored coins" can mean two different things:

- A technique for carefully constructing blockchain transactions in a way that preserves information apart from the base monetary value of the underlying units.
- The extra information preserved in the blockchain by employing the colored coin technique.

For the sake of clarity we will differentiate the technique from the information by using the term *virtual tokens* to refer to the extra information that is preserved using the colored coins technique.

II. Virtual Tokens

Virtual tokens possess all the capabilities of currency, plus one additional capability (smart property) which is helpful for non-currency usage.

Users of virtual tokens can:

- Transfer them between individuals
- Combine multiple tokens into a single token with a greater value
- Divide the value of a single token into multiple tokens
- Use them in blockchain-scripted contracts
- Store them on the blockchain with multisig scripts
- Unambiguously prove that any particular virtual token is a valid member of a set created by the issuer, without requiring the issuer to create and manage a token registry.

III. Enforcement

Virtual tokens represent ownership information, but they can't enforce real-world obligations. For example, a particular issuance of virtual tokens might represent tickets for entry to a concert. The virtual token can prove the bearer should be allowed to enter the concert, but it can't force the bouncer to step aside and let him pass. Colored coin techniques can't prevent the user from manipulating the underlying bitcoins in a way that destroys the extra information, because operations on virtual tokens are governed by Bitcoin transaction rules, and colored coin requirements are more strict but not enforced by the network. Using virtual tokens in a transaction that does not obey the colored coin rules destroys their extra meaning, leaving behind only their base monetary value. This is equivalent to taking one's paper concert ticket and setting it on fire.

IV. Metadata

The quantity and ownership of virtual tokens can be stored in the blockchain, but the semantic information that indicates what a token means is not (and cannot be) similarly stored. For example, the blockchain will track how many concert tickets have been issued and which address owns them, but not the fact that they represent authorised entry into a particular concert at a specific time and place. The storage of and operations on metadata require a specific kind of external system, such as Open-Transactions.

V. Blockchain Limitations

The speed of colored coin transactions, and the capabilities of scripted contracts that use virtual tokens are the same as those of the underlying blockchain.

12. Coloring Techniques

There are two techniques that may be used to create virtual tokens: transaction-based coloring and address-based coloring.

I. Transaction-based coloring

Transaction-based coloring was pioneered by ChromaWallet and works by identifying a specific Bitcoin transaction at a particular time as the "genesis transaction" and tracking all units which descend from the genesis transaction. Transaction-based coloring can produce the full range of virtual token types, and has the security property that even a loss of the original private keys to the genesis address cannot result in the issuing of counterfeit virtual tokens. This security property means the number of virtual tokens matching a color DeFinition is fixed at the time of creation and cannot be altered in the future—which can be an advantage or a disadvantage, depending on the application.

II. Address-based coloring

Address-based coloring was created by Coinprism and tracks bitcoins which are descended from any transaction that passes through a DeFined address. This means the issuer can easily create new units in the future, but so could a thief who manages to steal the private key for that address.

13. Virtual Tokens

The different use cases for virtual tokens can be divided into three general categories, which form the different types of virtual tokens.

I. Tickets

Tickets are transferable bearer tokens which are designed to be eventually redeemed for some kind of real world value.

Examples of tickets include:

- Event entry passes
- Store coupons and special offers
- Frequent flyer miles and other redeemable rewards

Both address-based and transaction-based coloring can be used to create tickets.

II. Certificates

Certificates are transferable and redeemable in the same manner as tickets, and they additionally entitle the bearer to some kind of revenue paid through the blockchain.

Certificates can be used for bearer securities, such as securitized loans, mortgages, bonds, and dividend-paying stocks.

Both address-based and transaction-based coloring can be used to create certificates.

III. Smart Property

Smart properties are transferable like tickets and certificates, and in addition, every particular smart property is both unique and atomic. Only one smart property of a given identifier can be

created, and once created it may not be subdivided.

Smart properties can be used to indicate ownership of a unique real-world asset, and can also be used for objective naming of content-addressed mutable data. This naming function is related to, and an extension of, hash-based naming.

A common operation in software engineering is to use cryptographic hash functions to create short identifiers for large pieces of data. This is useful because hash values are easy to communicate since they are short, and also are easy to check since they are deterministic. This means if you know the name of some piece of data, you can independently verify that you have the correct copy of it. But the limitation of hash-based naming is that the named data can never change.

Smart property overcomes this limitation. Because of a Bitcoin feature (OP RETURN) that allows arbitrary data to be attached to transactions, every time smart property is moved it can be associated with a new hash. This means if data is identified by a smart property identifier instead of using the hash, the identifier of the smart property can objectively and unambiguously identify the most current version of the data.

14. Example MatterToken-omics for a MatterFi Based Token Project

This section illustrates one of many possible hybrid on/off chain token projects that can be launched with the MatterFi ecosystem. MatterFi isn't a token company however we support many token ecosystems and almost all of our customers have a token.

Here we propose creating a set of Ethereum-based DeFi contracts (the "MatterConverter") that allow participants to stake crypto assets in order to earn accrued interest in return for supplying market liquidity. Borrowers, meanwhile can obtain loans from the staked assets but they have to provide collateral denominated in other asset types. The acceptable collateral is set by governance.

The MatterFi ecosystem is powered by a utility and governance token, the "MatterToken." In addition to this token, the MatterConverter also issues separate *"mTokens"* to users who stake funds. These mTokens are redeemable anytime for the underlying asset, and until that redemption occurs, the mTokens continue accruing interest.

For example if a user stakes some amount of ETH, the MatterConverter receives that ETH and in return, sends the user an equivalent number of mETH tokens, which are redeemable back anytime for actual ETH.

mTokens are redeemable at an exchange rate (relative to the underlying asset) that constantly increases over time, based on the rate of interest earned by the underlying asset. For example, the amount of ETH that can be exchanged for mETH increases every Ethereum block.

Users that either stake or borrow not only earn interest on their stakes, they also get rewards in the form of bonus MatterTokens utility tokens at launch. MatterTokens can be used for governance of the MatterConverter. This includes voting on proposed changes to the protocol.

Any user who possesses mTokens can post them as collateral via locking in order to borrow real crypto assets like ERC-20 tokens. When mTokens are locked the corresponding underlying asset isn't redeemable until the loan is paid off. To borrow the user must stake their collateral first. In the case of borrowing, the amount that must be repaid increases over time as interest accrues. If the borrower becomes insolvent before paying off his loan, then Liquidators are incentivized to pay off portions of the borrower's balance owed, in return for an increased percentage of the borrower's collateral. For example, if the liquidation incentive is 1.2, the MatterConverter pays liquidators an extra 20% of the borrower's collateral for every unit of debt that they close.

The MatterConverter is a fork of Compound. Thus far we have described features that are available in the open source code of Compound. The MatterFi equivalent of Compound's 'Compound token' is the MatterToken. The MatterFi equivalent of Compound's 'cTokens' is the mToken.

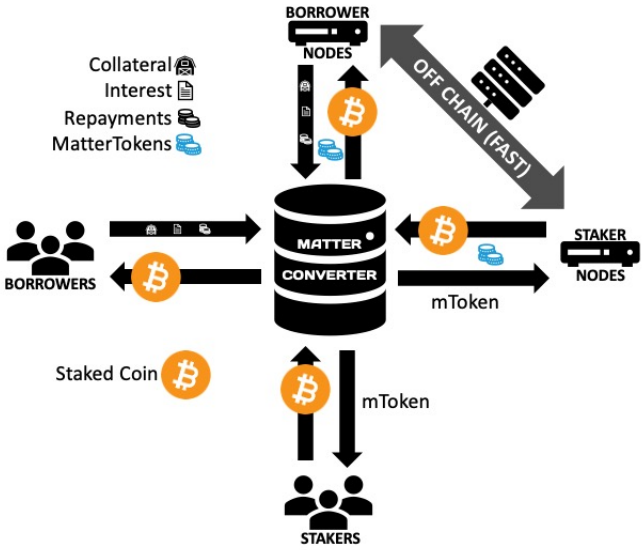
In this example, the MatterConverter could use the following patent pending additional elements to substantially differentiate from Compound:

- Hybrid On-Chain and Off-Chain MatterNodes. For faster, cheaper, and more powerful transactions, users can additionally deposit their coins and tokens into a MatterNode where off-blockchain transactions can occur based on cryptographic proofs. Users gain access to a variety of off-chain, cryptographically secure instruments such as digital cash, digital cheques, invoices, recurring payments, exchange, and DeFi smart contracts. MatterNodes can also organize into Pools, using blockchain-based multisig for secure storage of deposited funds, essentially providing their users with deposits that are safe from theft and fraud. MatterNode operators are rewarded with bonus MatterTokens for simply operating at a rate set by governance.
- MatterNode Transaction Fee Handling. MatterNodes earn transaction fees for processing off-blockchain transactions. At the point of each transaction, that fee is immediately sold for

MatterFi utility tokens, a percentage of which are then paid upstream to the MatterConverter. The percentage is set by MatterConverter governance. If a MatterNode operator isn't charging any transaction fees (charity, etc.) then they get to use our ecosystem for free. If they do charge transaction fees then we collect a portion of those for providing the software.

- MatterNode operators will receive increased staking rewards from the MatterConverter at a rate set by the governance. Users of MatterNodes will be able to enter into a staking contract with MatterConverter that's managed for them by the MatterNode. Therefore they can opt to receive mTokens that they can then use off-chain on the MatterNode.
- MatterNode operators will get to borrow from the MatterConverter at a discounted rate set by the governance.
- Our MatterID identity protocol enables on-blockchain DeFi contracts (and off-blockchain pools) to KYC-AML participants where appropriate. With our implementation, MatterFi and our customers could use the system to create financial instruments for entities with certain KYC-AML criteria. For example you could have an on-blockchain DeFi contract only for residents of the USA, and a different one for residents of Japan. As such we will be modifying Compound to add OBPP-05 capabilities to it with a patent pending OBPP-05 smart contract identity oracle. This will empower the MatterConverter with four new key capabilities controlled by governance:
  - OBPP-05 powered mToken issuance. A OBPP-05 enabled smart contract will know exactly who to send corresponding mTokens to when the contract receives a stake. This considerably simplifies smart contract operation as the smart contract no longer needs to accept a staker's receiving address blindly. This also eliminates the ability to use the smart contract for money laundering. With OBPP-05 the user is always treated as a single atomic entity by the smart contract as opposed to making the assumption that the user will always provide a receiving address that's really theirs. So when tokens are received from the sender when it's time to return them they will automatically go back to the original sender.
  - OBPP-05 enabled KYC/AML. The smart contract can be programmed to only accept funds according to jurisdictional rules. For example, a certain currency could be only available to OBPP-05 identities that have been KYC/AML'd by a certain entity. This is implemented by automatically refunding unapproved transactions where a user doesn't have a OBPP-05 with the signed credentials. Additionally, the claims may be blinded, see the "OBPP-05: Blind Claims" section above.
  - OBPP-05 smart contract audit-ability. Although 3rd parties can't divine anything meaningful about transactions between any specific user and the smart contract, the user and the smart contract can see the transactions between themselves when they use OBPP-05 as they know their own and the other parties receiving addresses. Thus a smart contract can maintain a log of transactions for some, all or none of its transactions that are tied to KYC/AML'd identities to meet jurisdictional standards and this information can be verified by inspecting the blockchain.

The system is depicted in the diagram below:



In this example, the daily token rewards will be distributed amongst the borrowers, stakers, and MatterNode operators. The remaining tokens will be placed in a reserve used for growth. As such the MatterConverter will be eventually controlled by the community and the reward rates and distribution shall be decided by an increasingly community controlled governance.

MatterToken is differentiated from other tokens in the utility it offers by allowing users to gain access to a hybridized crypto financial system where on-blockchain deposits are stored securely while off-chain instruments make transactions much faster, cheaper, and more powerful. Our 'MatterID' identity protocol would allow the MatterConverter liquidity reward system to interact with other on-chain decentralized-finance contracts and off-chain with KYC-AML when appropriate. This implementation would allow an on-chain contract to operate for residents of the US differently than one for residents of Japan, a fencing feature not previously available. Users would, for the first time, have the option of creating hybrid-crypto financial systems that they could tune to their real time needs.

15. Conclusion

The MatterFi system with its MatterID identity credentials, Open-Transactions, token issuance and voting pools represents a complete solution for the issuance, storage, and transaction processing of blockchain-based assets, creating interesting business use cases across a myriad of industries. For example:

- Retail Banking. Users can deposit cryptocurrencies in financial institutions (or federated collectives of financial institutions) operating pools. These "off chain" deposits are protected by the pool but don't require mining and process with speeds comparable to regular financial transactions. For example, a user could transact cryptocurrencies or even, a bank issued stable coin fiat equivalent.
- Exchanges, Investment, Trading Fintech. Voting pools are the currently most secure model for operating an exchange that can trade any asset. Furthermore, the identity credentials afforded by OBPP-05 allow money transmitters to KYC/AML their users and have cryptographic proof that they are sending and receiving funds from identities they verified. This solves big problems in meeting compliance for centrally regulated businesses in a global decentralized ecosystem.
- Token Offerings. MatterFi's Colored Coin implementation facilitates easy issuance and liquidity of any type of token, be it a security, asset, or "coin." Token issuers with the MatterFi system can make cross-chain moving of tokens possible. For example, start off on Ethereum with ERC-20 tokens and move them to the Bitcoin Cash blockchain at the user's request. Tokens on disparate blockchains can also be deposited into a single OT MatterNode and traded against each other.
- Gaming/Social Media. Presently these ecosystems usually employ their own in-house financial transaction systems that aren't globally liquid. If they use cryptocurrencies for payment, which is commonplace, the present solutions pose money laundering challenges as game operators find themselves unwilling middle men for illicit transactions. MatterFi's identity scheme makes it easy for game companies to digitally meet the same financial requirements as traditional fintech's while maintaining their business model.

Presently and in the foreseeable future, global economies rely on centralized "value creators" offering services which is why users use these services. The fundamental problem MatterFi solves is bridging the gap between the modern value creation system and its consumers, where everyone desires to do so in a decentralized global environment supporting thousands of currencies. MatterFi hopes to create a worldwide standard for off-blockchain and on-blockchain financial functionality that allows centralized value creators to do business in a globally decentralized crypto currency distributed finance system.

16. High-level DeFinitions

I. Blockchain

A *blockchain* is a hash-chained series of blocks comprising a ledger. Each block contains a series of transactions (aka ledger entries), as well as the hash of the previous block in the chain. An attacker can never change any existing block, because that would necessarily give it a different hash, and since that hash appears in the next block, all subsequent blocks would be affected. The proof-of-work necessary for such an attack is untenable; it would impose an exorbitant cost on the fraudulent miner and in any case, would not gain him anything.

The biggest, most well-known blockchain, Bitcoin, exists as a peer-to-peer network, where each peer maintains its own, identical copy of the blockchain ledger. The peers are able to agree on the state of the ledger based on the axiom that the longest proof-of-work chain is the "true" one. Anyone can pop up anonymously and start mining (that is, processing transactions), and earn an honest living as long as he doesn't try to defraud the network. Anyone who attempts to defraud the network will not succeed, and he will also throw away a lot of money in the process, due to the proof-of-work requirement for Bitcoin mining.

Since miners in Bitcoin are anonymous, they are also permissionless. There is no central authority with the power to decide who can mine and who can't. This "miner anonymity" is why Bitcoin is sometimes called a "permissionless" chain. In contrast, other protocols, such as Ripple/Stellar, make use of some central authority with the power to choose who can process transactions, and who cannot. Since these systems require centralized permission to authorize their validators, such systems are called "permissioned" for that reason. Due to this difference, they are also often referred to as "distributed ledgers" instead of as "blockchains". Put another way, a blockchain is a kind of distributed ledger. Specifically, the kind that is permissionless and requires proof-of-work. Whereas Ripple/Stellar is permissioned and uses a different protocol for achieving consensus instead of proof-of-work. Ripple is not a blockchain, but it is still a distributed ledger.

Another family of blockchains is based on the Ethereum network. The differentiator is that Ethereum scripts are Turing-complete, whereas Bitcoin scripts are Turing-incomplete. This basically means that Ethereum scripts are allowed to have loops in their code, and potentially infinite loops. Bitcoin disallows loops, because infinite loops would cause the mining rigs to freeze up for eternity. Ethereum solves this problem by allowing users to continually add "gas money" to a running script to pay for its continued operation. Ethereum miners are happy to continue running the script–even one containing an infinite loop–as long as they will continue to be paid for running it.

II. Coin

A *coin* is the base currency of any distributed ledger or blockchain. The best example of a coin is Bitcoin itself. Most other alt-coins are simply forks of the Bitcoin code. (They're all mostly the same under the hood). Examples include Litecoin, Bitcoin Cash, Dash, Dogecoin, etc.

Each of those blockchains has its own base currency. The base currency of the Bitcoin blockchain is BTC. The base currency of the Litecoin blockchain is LTC. The base currency of Bitcoin Cash is BCH, and so on.

The other distributed ledgers, such as Ripple, Stellar, and Ethereum, also have their own base coins (XRP, XLM, and ETH, respectively).

III. Token

A *token* is a new unit type that can be issued onto a blockchain and circulate on that chain just as the base coin circulates. There are different protocols for doing this. For example, the ERC-20 token protocol is used for issuing tokens onto the Ethereum blockchain. Similarly, the SLP token protocol is used for issuing tokens onto Bitcoin-family blockchains. For example, SLP could be used on Bitcoin Cash, or Litecoin, or Dash, etc., to issue tokens onto those blockchains.

IV. Stablecoin

A *stablecoin* is a token that is supposedly redeemable 1-to-1 for some national currency (such as the dollar) by an issuer. A good example of this is Tether, who issues a dollar token and who supposedly possesses enough dollar reserves to redeem every one of their tokens in circulation. We say 'supposedly' because it is the user's responsibility to do his own due diligence on each dollar issuer, and to decide which issuers he will trust.

FINCEN recognizes "convertible virtual currencies" as any virtual currency that is convertible back into "real" currency, or that acts as a substitute for "real" currency.

FINCEN regulates any "virtual currency administrator" who has issued such a token and requires the issuer to have a money transmitter's license and to be AML/KYC compliant.

V. Security Token

A *securitytoken* is a token that represents some type of security. For example, a token may be redeemable for some commodity (such as gold), or it may represent stock in a company. Any situation where the holder is promised some future return as a result of "future efforts of the team" also falls into this category.

Security tokens are regulated by the SEC.

VI. Utility Token

A *utilitytoken* in the current day and age is usually considered by investors as "any token that isn't a security." However, the classical interpretation of the term usually includes the fact that the token is somewhere redeemable for some utility provided by the network. For example, Factoids are a utility token that users have to use in order to pay for storing data proofs into the Factom blockchain.

Another example of a utility token would be one that powers an ATM network. Which is to say, any transactions occurring on any of those ATMs would subtract a small transaction fee in the local currency and convert it to ATM tokens in order to pay for that ATM transaction.

Having some real-world utility tied to the token ensures that they will always have some minimum real-world value.

VII. Fundraising

Security tokens and utility tokens can also be used for fundraising, since the issuer can print up a number of new units and then, if there is demand for them, sell them and create market liquidity. The issuer can continue selling into growing demand, using his issued tokens to fund his operations and pay his employees. In these circumstances, the increasing market value of the token becomes the overriding concern.

This activity is regulated by the SEC wherever these tokens are sold to U.S. Persons.

VIII. Cross-chain Tokens

It's possible for an issuer to create units of his tokens on multiple blockchains. For example, he may issue 100,000 units onto the Bitcoin Cash blockchain using the SLP protocol, and then issue another 100,000 units onto the Ethereum blockchain using the ERC-20 protocol. In this example, there are a total of 200,000 units in circulation, half on each chain.

This makes it possible for users to move their tokens from one blockchain to another. The issuer (or any other market maker) simply offers to trade the tokens 1-to-1, enabling users to convert tokens on one chain into tokens on another. In the user's experience, this conversion can also be managed "behind the scenes" by his wallet software, preventing any need for the user to consciously have to place market offers whenever he desires to move his tokens from one chain to another.

We note that cross chain tokens have yet to be created by anyone. We posit that this is true because there hasn't been a reliable cryptographically secure off chain system to move a token between chains.

References

[1] David Chaum. *Blind Signatures for Untraceable Payments*. In D. Chaum, R.L. Rivest, and A.T. Sherman, editors, *Advances in Cryptology Proceedings of Crypto 82*, pages 199–203, 1983.

[2] Jonald Fyookball, James Cramer, Unwriter, Mark B. Lundeberg, Calin Culianu, and Ryan X. Charles. *SLP Token Type 1 Protocol Specification*, 2018. `https://github.com/simpleledger/slp-specifications/blob/master/slp-token-type-1.md`.

[3] Ian Grigg. *The Ricardian Contract*. In *Proceedings of IEEE Workshop on Electronic Contracting July 6*, pages 25–31, 2004.

[4] Ian Grigg. *Triple Entry Accounting*, 2005. `http://iang.org/papers/triple_entry.html`.

[5] Ben Laurie. *Lucre: Anonymous Electronic Tokens v1.8*. Technical report, 1999, 2008.

[6] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*, 2008. `http://bitcoin.org/bitcoin.pdf`.

[7] Chris Odom. *Open-Transactions: Secure Contracts between Untrusted Parties*, 2010. `http://www.opentransactions.org/open-transactions.pdf`.

[8] Chris Odom. *Triple Signed Receipts*, 2010. `http://opentransactions.org/wiki/index.php?title=Triple-Signed_Receipts`.

[9] Justus Ranvier. *Payment Codes*. `https://www.reddit.com/r/Bitcoin/comments/3alzga/bip47_reusable_payment_codes/`.

[10] Justus Ranvier. *Voting Pool Deposit Process*. `http://opentransactions.org/wiki/index.php/Voting_Pool_Deposit_Process`.

[11] Justus Ranvier. *Voting Pool Withdrawal Process*. `http://opentransactions.org/wiki/index.php/Voting_Pool_Withdrawal_Process`.

[12] Justus Ranvier. *Voting Pools*. `http://opentransactions.org/wiki/index.php/Voting_Pools`.

[13] Justus Ranvier. *BIP-47 Specification*, 2015. https://github.com/bitcoin/bips/pull/159.

[14] Justus Ranvier. *Universal Identity*, 2015.

[15] Justus Ranvier. *OBPP-05 Specification*, 2021. https://github.com/OpenBitcoinPrivacyProject/rfc/blob/master/obpp-05.mediawiki.

[16] Justus Ranvier and Jimmy Song. *Hierarchy for Colored Voting Pool Deterministic Multisig Wallets*. https://github.com/Open-Transactions/rfc/blob/master/bips/bip-vpc01.mediawiki.

[17] Justus Ranvier and Jimmy Song. *Hierarchy for Non-Colored Voting Pool Deterministic Multisig Wallets*. https://github.com/Open-Transactions/rfc/blob/master/bips/bip-vpb01.mediawiki.

[18] Bill St. Clair. *Truledger in Plain English*, 2008. http://truledger.com/doc/plain-english.html.