GLOBAL ASSENT TOKEN TECH WHITE PAPER

1. Purpose

The goal of this project is to provide the world's first black empowerment crypto token to be used in business, non-profits, and within black families everywhere. We call it *Global Assent Tokens*, shown as "GAT" on the exchanges.

2. Scope

We are a digital currency platform that embraces and supports the economic systems of the black community. *Global Assent Tokens* can be utilized by logging into the web dashboard wallet. This dashboard wallet interface can be used on all devices through the web interface or as a decentralized application built on the ethereum platform, users will have the option of both. We will create a web dashboard wallet as version 1, an ethereum based system agnostic application as version 2, and then we will deploy an original blockchain implementation for version 3 of our token.

3. Product Perspective

3.1 System Interfaces:

Public GAT Token users can use our custom dashboard to make transactions or they can use a third-party browser wallet called MetaMask. MetaMask allows users access to the web3.js api and a connection to the an Ethereum wallet, and importantly the ability to use the Ethereum wallet to sign transactions on the Ethereum blockchain/ network.

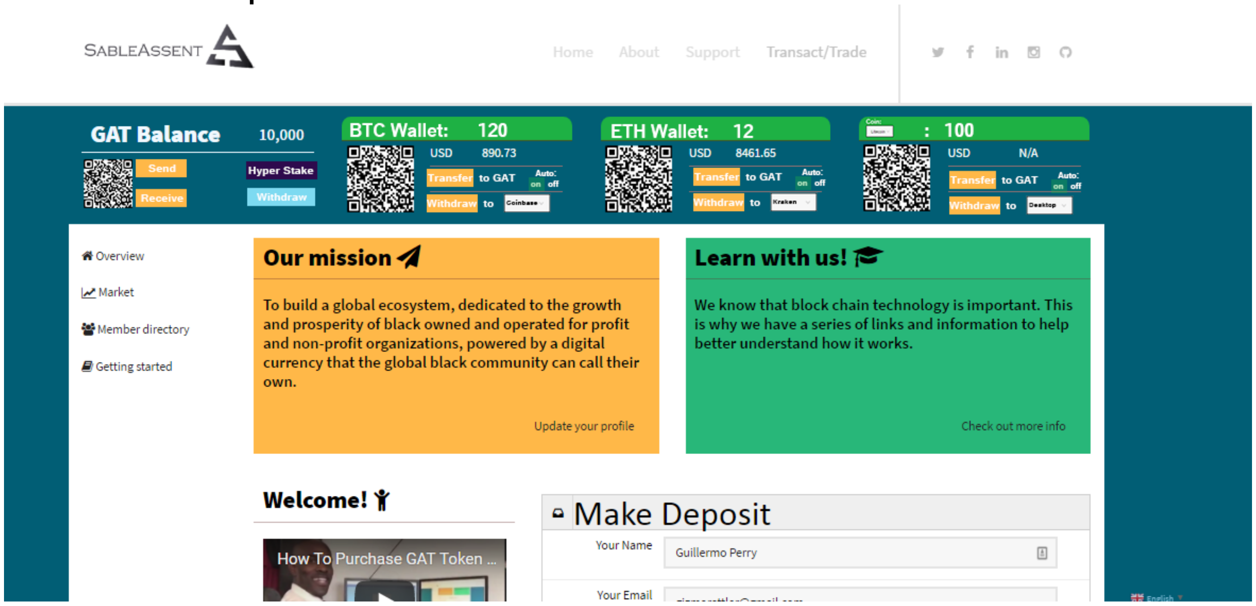
a. The Ethereum client would be part of the web browser.
b. The blockchain network would be our private consortium on blocks located on our amazon web cloud linux server. The user will have to be verified in order to create their own node, or connect to the central server.
c. Our structure will look something like this:

[Web Browser *(end user)***]** <==> **[Server** *(web application/javascript <==> web3 <==> ethereum client)***]** <==> **[Ethereum Network** *(solidity code)***]**

3.2 User interfaces:

A first-time user of the dashboard will opt to sign in from the main SableAssent.com login page. Once logged in, the user will see their dashboard page with navigation links on the left. Every user will be able to edit their personal dashboard details on the profile page.

General User Example Interface:



Our mission

To build a global ecosystem, dedicated to the growth and prosperity of black owned and operated for profit and non-profit organizations, powered by a digital currency that the global black community can call their own.

Update your profile

Learn with us!

We know that block chain technology is important. This is why we have a series of links and information to help better understand how it works.

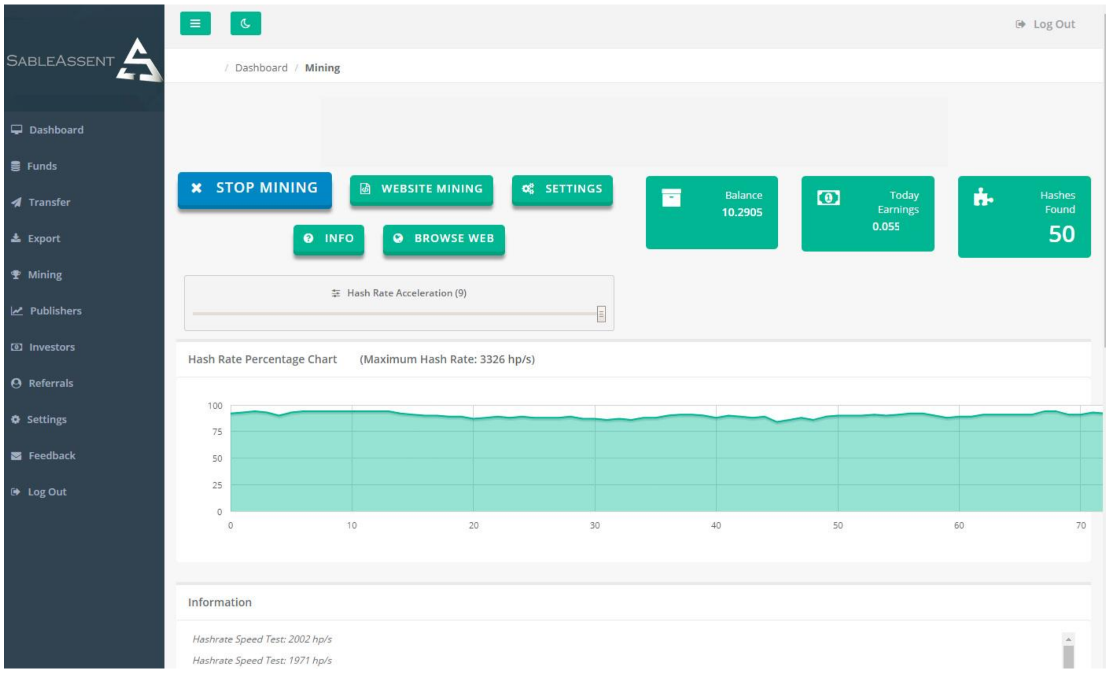
Check out more info

Make Deposit

Your Name: Guillermo Perry

Your Email

Token Ambassador:



Balance 10.2905

Today Earnings 0.055

Hashes Found 50

Hash Rate Percentage Chart (Maximum Hash Rate: 3326 hp/s)

Information

Hashrate Speed Test: 2002 hp/s

Hashrate Speed Test: 1971 hp/s

3.3 Hardware Interfaces:

All users will have the option of securing their dashboard wallet keys with physical token wallet storage devices called Hardware Wallets. Here are two examples:

a. Ledger wallet - https://www.ledgerwallet.com/
b. Digital Bitbox - https://www.cryptohwwallet.com/digitalbitbox.html

Token Ambassadors are required to purchase a physical node in order to insure hardware consistency across the entire network. Please click here to see the entire Token Ambassador rig proposal: https://sableassent.com/token-ambassador-rig/

3.4 Software Interfaces:

3.4.1 Ambassador Desktop Variation

Only Token Ambassador Users that opt to administrate and run their own node will then be required to download an entire copy of our private blockchain and therefore would be required to install "Go Ethereum", also known as "Geth", which will manage the local instance of the block chain and enforce consensus among the many nodes.

a. Geth package installation can be found here: https://geth.ethereum.org/install/
b. This optional interface would allow the dashboard wallet application to operate in a decentralized manner. This interface is not required in the initial web wallet release of the application.

3.4.2 Internal digital Currency transaction systems

We will need to be able to change user currency and coins into ETH and GAT tokens. Furthermore, GAT tokens will be in the form of two separate tokens, one for unverified users on the public Ethereum blockchain, the other is for verified users on our private chain. We will use these software APIs to exchange tokens internally:

a. https://www.bitgo.com/
b. https://www.shapeshift.io/

3.5 Communications Interfaces:

Web wallet users will communicate with our central server over https protocol. Node users will have geth installed on their local systems. Geth uses RPC and IPC protocols to communicate between nodes for data duplication/verification aka consensus.

3.6 Memory Constraints:

The web wallet will only use as much RAM as the normal browser uses. Geth will determine how much memory is used on the client side if the user is required to run a node.

3.7 Operations:

User Story specifications

Actor-Goal list

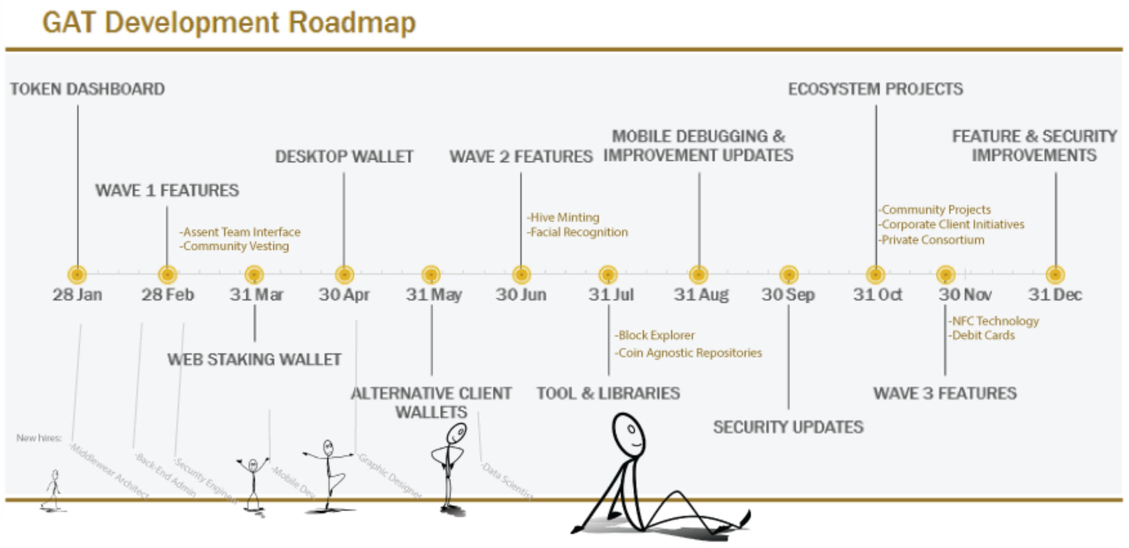
Actor	Goal

Token Ambassador (Decentralized Application & Node Administrators)	1. Verify all vendors. (Add, remove, modify). 2. Receive token (GAT). 3. Send token (GAT). 4. Mint new tokens (GAT) for Sable Assent. 5. Deposit currency. 6. Withdraw currency.
Vendors (verified token users)	1. Receive token (GAT). 2. Send token (GAT). 3. Exchange tokens with Sable Assent. 4. Publish news items. 5. Follow token (GAT) status. 6. Verify Identity with Ambassadors. 7. Deposit currency. 8. Withdraw currency.
Assent Team (Self-Employed Service Providers)	1. Review and sign Vendor project agreements. 2. Receive token (GAT). 3. Send token (GAT). 4. Exchange tokens with Sable Assent. 5. Follow token (GAT) status. 6. Update vendor marketing project status. 7. Deposit currency. 8. Withdraw currency.

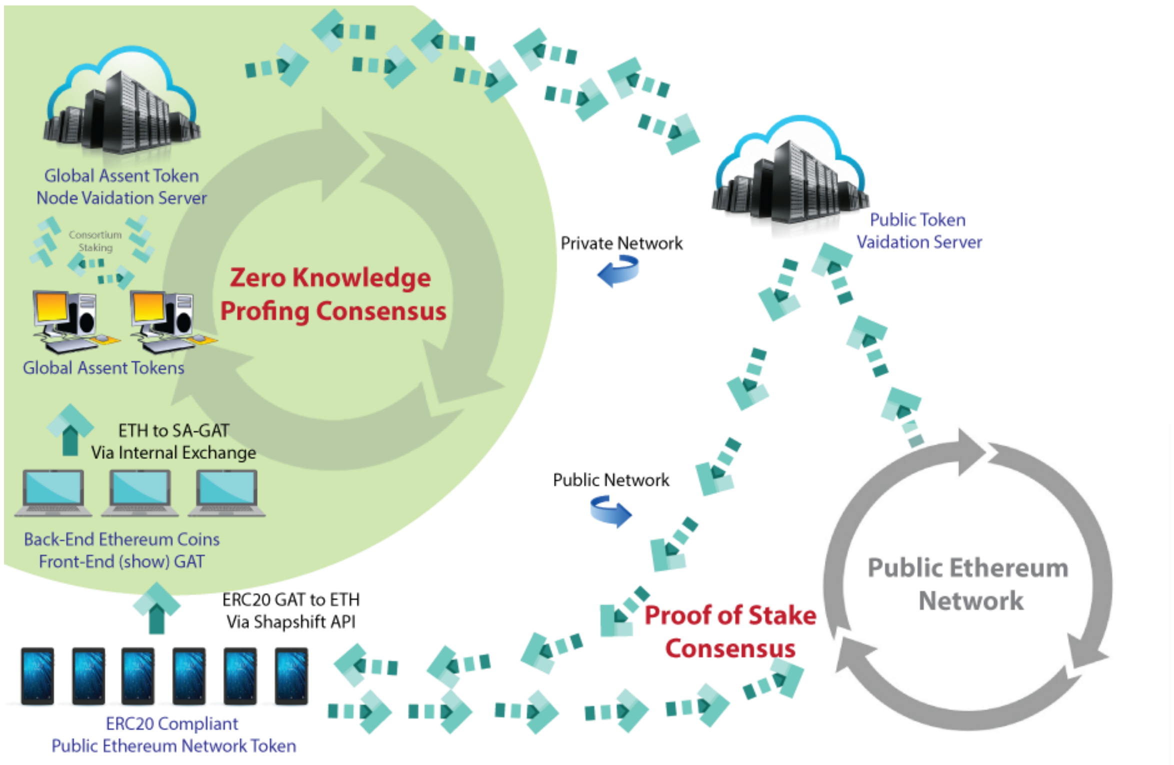
Token Holders (Non-verified token users)	1. Receive token (GAT). 2. Send token (GAT). 3. Exchange token with Sable Assent. 4. Follow token (GAT) status. 5. Review and sign Sable Assent terms and agreements.

3.8 Site Adaptation Requirements

The web wallet will be the initial focus for launch. We will add the Token Ambassador feature of node operations once the web wallet is up and running. Also, desktop and mobile versions of the light app will roll out throughout the year as specified in the development roadmap:



4. Product Functions:



----

5. User Characteristics

Token Ambassadors - Decentralized network of GAT network administrators.
Corporate Vendors - Independent business owners.
Assent Team - Self Employed service providers.
Token Holders - Unverified users and holders of the public GAT token.

6. Limitations

Gas consumption

Ethereum uses the concept of gas which means the sender of a transaction needs to pay (i.e. Eth or Etc) for the computational steps executed by the smart-contract that is invoked by the transaction. The more complex computations the smart contract executes, the more gas will be consumed. Therefore the transaction specifies a gas limit and a gas price.

Gas limit is there to protect you from buggy code running until your funds are depleted. The product of *gasPrice* and gas represents the maximum amount of “Wei” that you are willing to pay for executing the transaction. What you specify as *gasPrice* is used by miners to rank transactions for inclusion in the blockchain. It is the price in Wei of one unit of gas, in which VM operations are priced.

Determining the right gas-consumption is crucial for correct functioning of Ethereum. Too low gas introduces a DOS vulnerability, attackers can make the network slow by calling computationally hard functions while paying relatively little. Too high gas wastes people’s money.

7. Assumptions and Dependencies

One fundamental concern in blockchain technology is the confidentiality of the data on the blockchain. In order to reach consensus between all independent nodes in a blockchain network, each node must be able to validate all transactions (for instance against double-spent), in most cases this means that the content of the transactions is visible to all nodes. Fortunately several solutions exist that preserve confidentiality on a blockchain (private transactions, HyperLedger Fabric Channels, Payment Channels, Homomorphic encryption, transaction-mixing, zero knowledge proofs etc.).

Our solution is to utilize the Proof-of-Stake protocol where only Token Ambassadors will have the option of depositing cash into the system in return for a guaranteed stake in the monthly dispersement of token transaction fees. **In this way, no mining is necessary because we will instantiate the value of the token with Ethereum deposits and transaction fees.**

Please visit us here for our Staking Algorithms & Calculations: https://sableassent.com/staking-algorithm-calculations/

The Token Ambassador will be responsible for a full node. This includes verifying every new vendor application that comes onto the network. The Token Ambassador deposits $10,000 into the network and has vested interest in assuring that every vendor is fully verified. If verification is not done correctly the Token Ambassador will have a small portion of their stake deducted.

8. External Interfaces

Decentralized computation platform: https://www.truebit.io/
Blockchain information APIs: https://blockchain.info/api/charts_api
https://www.cryptocompare.com/api/#-api-data-coinlist-

9. Functions

This is a summary of the major functions the app will perform. Most of the major tokens on the Ethereum blockchain are ERC20-compliant. The ERC20 token standard describes the functions and events that an Ethereum token contract has to implement in order to be widely accepted by wallets and exchanges everywhere.

Following is an interface contract declaring the required 6 functions and 2 events to meet the ERC20 standard:

```
pragma solidity ^0.4.0;
contract ERC20 {
    function totalSupply() constant returns (uint totalSupply);
    function balanceOf(address _owner) constant returns (uint balance);
    function transfer(address _to, uint _value) returns (bool success);
    function transferFrom(address _from, address _to, uint _value) returns (bool success);
    function approve(address _spender, uint _value) returns (bool success);
    function allowance(address _owner, address _spender) constant returns (uint remaining);
    event Transfer(address indexed _from, address indexed _to, uint _value);
    event Approval(address indexed _owner, address indexed _spender, uint _value);
} // https://github.com/ethereum/EIPs/issues/20
```

Our token will include further information describing the token contract:

```
--------------------------------------------------------------------------------
string public constant name = "Global Assent Token";
string public constant symbol = "GAT";
uint8 public constant decimals = 9;  // 18 is the most common number of decimal places
```

We will also implement the ERC223 Standard

3 Primary improvements with ERC223:

(1) **Eliminates the problem of lost tokens** which happens during the transfer of ERC20 tokens to a contract (when people mistakenly use the instructions for sending tokens to a wallet). ERC223 allows users to send their tokens to either wallet or contract with the same function transfer, thereby eliminating the potential for confusion and lost tokens.
(2) **Allows developers to handle incoming token transactions**, and reject non-supported tokens (not possible with ERC20)
(3) **Energy savings.** The transfer of ERC223 tokens to a contract is a one step process rather than 2 step process (for ERC20), and this means 2 times less gas and no extra blockchain bloating.

Switching from ERC20 to ERC223
"ERC23 tokens are backwards compatible with ERC20 tokens. It means that ERC23 supports every ERC20 functions and contracts or services working with ERC20 tokens will work with ERC23 tokens correctly."
Sources: https://github.com/Dexaran/ERC23-tokens/tree/Recommended#erc23-token-standard(ERC23 and 223 are the same thing). To read the full discussion about ERC20 and ERC223, follow this link: https://github.com/ethereum/EIPs/issues/223

Custom Functions:

Here is a list of custom functions that need to be created by our team and implemented into the GAT dapp. These functions do not necessarily need to be in the token's smart contract because it would then cost gas to perform these essential actions.

1. Facial Recognition
2. Near Field Communication Tech
3. Community Staking
4. Automated Internal Exchange from Public GAT to Private GAT

Extended feature functions will be put together by the development team during brainstorming meetings. The main functions have been described in section 4 of this document.

10. Verification

a) **Form Validation** - There are different types of form validation that you'll encounter on the web:

- Client-side validation is validation that occurs in the browser, before the data has been submitted to the server. This is more user-friendly than server-side validation as it gives an instant response. This can be further subdivided:

    - JavaScript validation is coded using JavaScript. It is completely customizable.
    - Built in form validation is done with HTML5 form validation features, and generally doesn't require JavaScript. This has better performance, but it is not as customizable.
- Server-side validation is validation that occurs on the server, after the data has been submitted — server-side code is used to validate the data before it is put into the database, and if it is wrong a response is sent back to the client to tell the user what went wrong. Server-side validation is not as user-friendly as client-side validation, as it requires a round trip to the server, but it is essential — it is your application's last line of defense against bad (meaning incorrect, or even malicious) data. All popular server-side frameworks have features for validating and sanitizing data (making it safe).

b) **Key Validation** - GAT's ETH CryptoNote validation solves the problem of tracing transactions and duplicating keys by implementing automatic unique one-time keys, derived from the public key, for each p2p payment. This innovation is derived from a modification of the Diffie-Hellman exchange protocol. Originally it allows two parties to produce a common secret key derived from their public keys. In our version the sender uses the receiver's public address and his own random data to compute a one-time key for the payment. The sender can produce only the public part of the key, whereas only the receiver can compute the private part; hence the receiver is the only one who can release the funds after the transaction is committed. He only needs to perform a single-formula check on each transaction to establish if it belongs to him. This process involves his private key, therefore no third party can perform this check and discover the link between the one-time key generated by the sender and the receiver's unique public address.

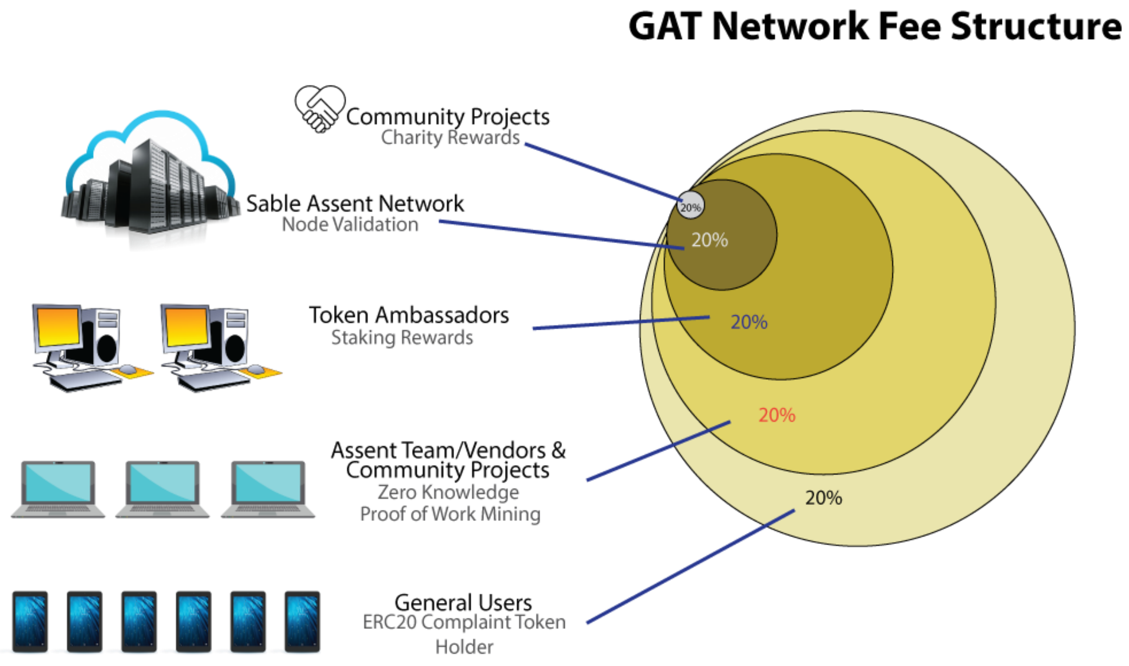
c)**Negative Proofing** - Since the main blockchain we are using is private, whether it is used only by us internally or by multiple parties, it will be anchored into the Ethereum blockchain periodically to prove it wasn't altered in any way (Factom does this for example). Once we have that, we would also need to periodically archive a complete copy of the blockchain (or at least the relevant slice between two anchors) as part of a future audit. If it is our internal blockchain, it would be analysed in whole, if it is shared - we would need to indicate which parts we used just like in the public blockchain scenarios. We will then create our final data compilation for our audit, consisting of:
- The entire block history in the slice of time we are analysing.
- Whatever else is needed to prove the block history was unaltered. This can come in block header chain up to the newest Bitcoin block, simplified-payment-verification-esque branches of anchor transactions included in blocks, etc..
- Our original commits to the addresses we would use (if applicable), along with the necessary proofs that we committed to them at the appropriate time.
- Any relevant metadata we wish to submit (descriptions of which transaction was for what, etc.)

Finally, we would have not only a cryptographically verifiable proof that all of the transactions took place, but also have irrefutable proof of the time frame they took place in (we couldn't forge a few extra transactions from last year after the fact) and be able to prove that we didn't omit any piece of data - creating a negative proof.
The last one is possible because the records we are dealing with are cryptographically sealed (we can't alter the blockchain without invalidating its future, which would be

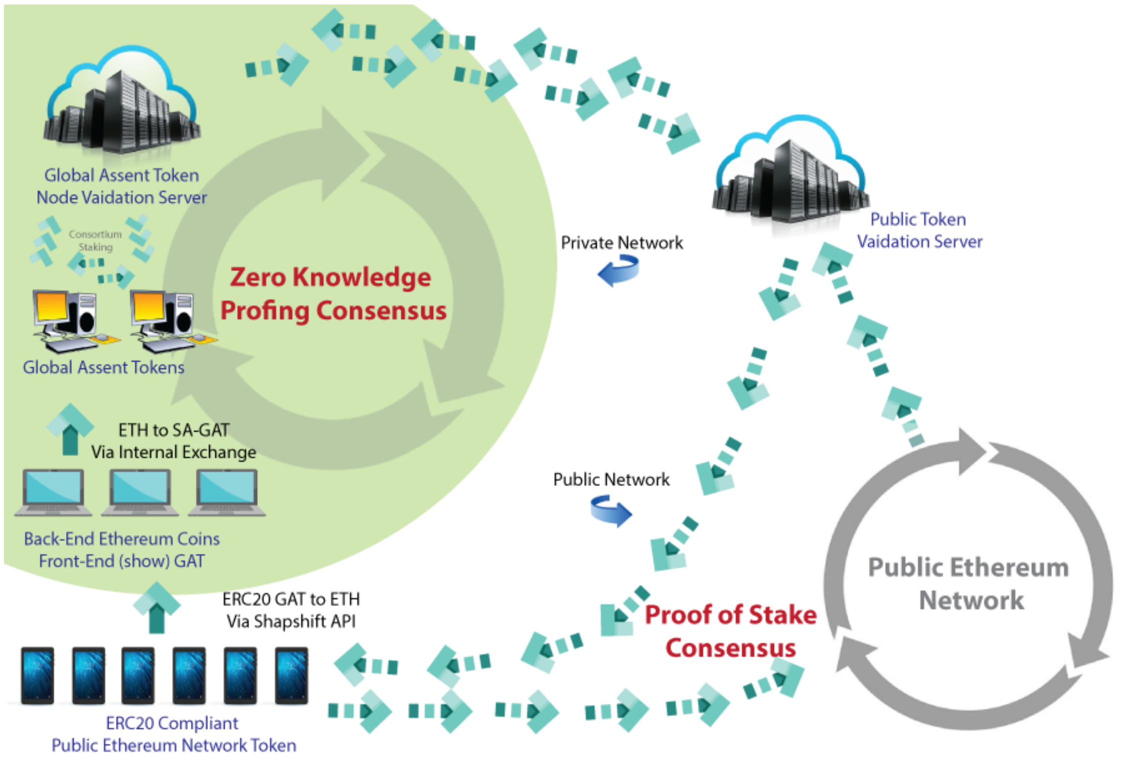
evident), but also public and finite (we CAN iterate over every block and every transaction and check whether it is relevant to the audit or not). This way we not only provide every relevant transaction, but prove there are no relevant transactions we didn't provide.

Thanks to the advent of cryptography and blockchain technology with atomic, countable transactions, it is now possible to create undeniable cryptographic provable complete audits. Hopefully this will help us avoid more audit fraud cases in the future.[2]

11. Supporting Documentation



Network Structure:



12. The Future

Once we have established our community using the Ethereum platform, we will begin creating our own block chain that operates independent of any other platform. We will seek this avenue as security becomes increasingly important with the growth of the network.

The zero-knowledge range proof allows the blockchain network to validate that a secret number is within known limits without disclosing the secret number. This is useful to reach consensus in a variety of use cases:

Validate that someone's age is between 18 and 65 without disclosing the age.
Validate that someone is in Europe without disclosing the exact location.
Validate that a payment-amount is positive without disclosing the amount (as done by Monero).
The zero-knowledge range-proof requires a commitment on a number by a trusted party (for instance a government committing on someone's age), an Ethereum-user can use this commitment to generate a range-proof. The Ethereum network will verify this proof.

What is Zero Knowledge Proof?

As explained in ING's white paper:

> *A zero-knowledge proof (ZKP) is a cryptographic method that allows a party (the prover) to prove to another party (the verifier) that a given statement is true, without conveying any additional information. For example, a ZKP allows you to proof that you're of a certain age, without revealing your actual age. To make it less abstract, there is an easy to understand demonstration given by Chalkias and Hearn [2]. The idea is as follows. Imagine you have a color-blind friend. Your friend has a red and a green ball that are otherwise identical. Your friend is not sure if these actually differ in color. Your job is to convince your friend that these balls are different in color, and reveal nothing else. The proof works as follows. Your friend puts these balls behind his back. Then, he brings the balls forward, shows them to you, and returns the balls to his back. You now know in which hand is the red, and in which hand is the green ball. Then your friend may switch the balls behind his back, and shows you a single ball (after putting it back again), and asks you: Did I switch the balls? Remember, your job is to convince your friend that the balls are different in color. Now, consider for a moment that the balls are of the same color. Then you have a 50% of guessing correctly if the balls were switched. After, say 1000 switches, the Law of Large Numbers [3] states that you have guessed approximately 50% of the switches correct. However, if the balls are of different color, you would end up with a very high number of correct statements (over 99%), since you can determine if the switch has been made. Since obtaining such a high score is highly unlikely when the balls are of the same color, your friend can assume that you are stating the truth: the balls are indeed of a different color. From this example we learn that there are ZKP has three properties: 1. Completeness. If the statement is true, an honest verifier will be convinced by an honest prover. 2. Soundness. If the statement is false, no dishonest prover can convince an honest verifier. 3. Zero-knowledge. If the statement is true, no dishonest verifier learns anything other than the fact that the statement is true. [1]*

Resources
[1] Retrieved from: https://www.ingwb.com/media/2122048/zero-knowledge-range-proof-whitepaper.pdf

[2] Retrieved from:
http://tpbit.blogspot.com/2016/01/positive-and-negative-proofs-in.html

-Written and compiled by Guillermo Perry