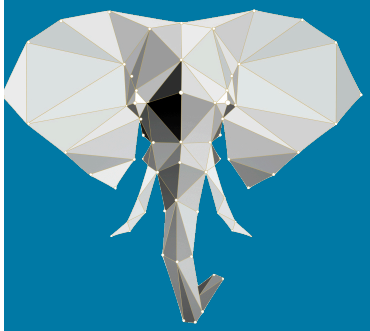


Plutus International

Plutuscoin WhitePaper

Version 1a – Plutuscoin v0.5.0

Abstract

EON is a decentralized blockchain based-platform that provides an infrastructure for th Exscudo Ecosystem services. This platform is not meant to be viewed as an alternative to Bitcoin, Ethereum and other altcoin, as it is designed for the execution of specific tasks of Exscudo. This fact determines the decisions made. For the realization of the platforms, approaches that have been tested in other existing crypto platforms are being used.

Introduction

Bitcoin gave us enthusiasm and an understanding of decentralized solutions. Since then many developers and teams have tried and succeeded in modernizing the source code ir order to advance and enhance the capabilities of the protocol. We are not an exception. We strived to create a fast and flexible decentralized network that would be able to realize business scenarios that require either trust or escrow agents. Besides, in the face of the blockchain's growing volume problem (scalability), our blockchain must not become enormous. This will enhance the possibilities of maintenance of the decentralized network. Moreover, the rush for the right to sign a block must not be hardware-dependent. It should fit the model and logic of the network in a much deeper manner.

EON solves all these tasks. The transaction is flexible and fast (1333 transactions per minute)[1] . We can make the blockchain 'collapse onto itself' an infinite number of time without losing the integrity of the network. This is an unrivalled innovative solution tha allows us to keep the size of the operational blockchain limited. The architecture of the platform is built on a simple core that realizes a mathematical model and services that provide additional functionalities. The core forms the decentralized part of the system that consists of a variety of peers and executes the functionality of support on user account and financial operations. Every peer stores the full information and carries out the validation of transactions task. When the peer completes this task, he receives a reward. Practically, the peer receives a commission when the block is created, and it equals the sum of commissions for all transactions in that block.

The services, on their side, interact with the distributed part and can receive a margin for offering some functionality. These services can be centralized or decentralized objects. This classification allows us to keep the core as simple as possible. It provides for additional options of development and growth of external services independently from the network core. The question of trust on external services which arises when the functionality doesn't work 'on one core', is solved for every single service in particular

Token Infrastructure

Plutuscoins are the internal digital asset of Plutus and are issued on the litecoin's blockchain. Plutuscoin trans- actions can be sent and received using the Plutus Mobile App, similarly to Bitcoin. Plutuscoins have the added benefit of zero transaction fees, as well as instant transaction confirmations.

Ownership of Plutuscoins is tied to an Litecoin account. Plutuscoin token source code — running on the production Litecoin network (networkid 1) — acts as a decentralized and distributed ledger of all Plutoscoin transactions and the addressed location of every Plutuscoin. The benefits of using Litecoins

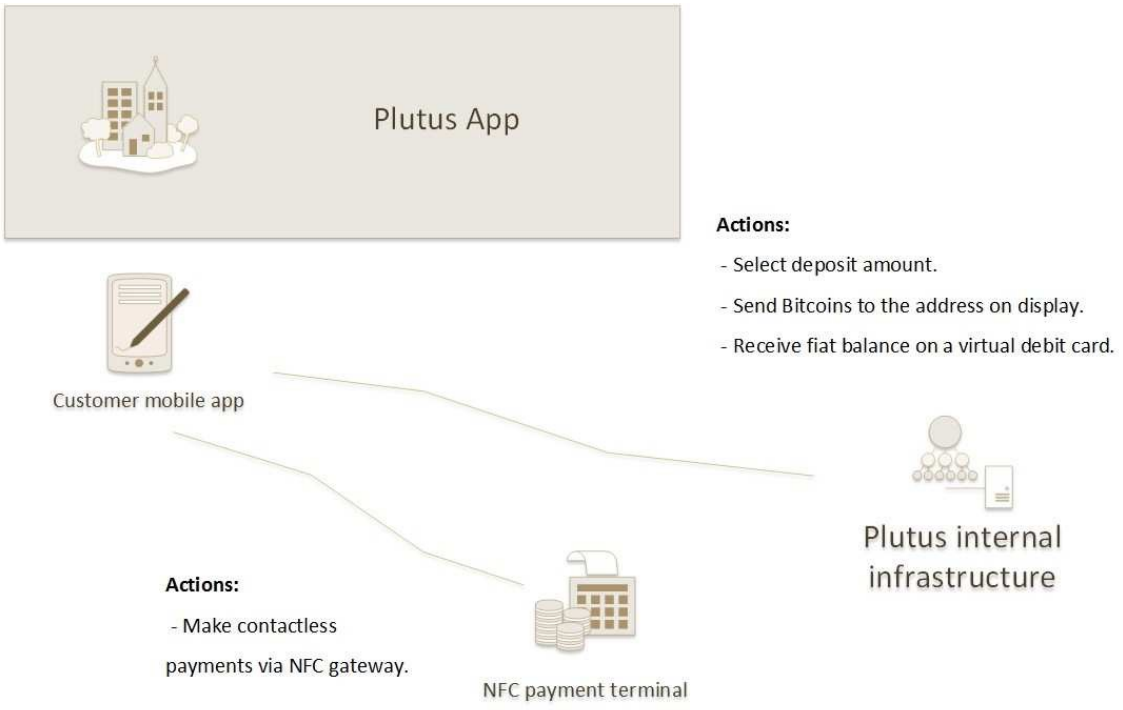
reliability, underwritten by Litecoins built-in incentives for securing the blockchain. This ensures the continuity, integrity and security of the Plutuscoin ledger.

The Litecoin network supports near-instant confirmations of Plutuscoin transactions. According to the latest data, the average time for a new transaction to be confirmed on the blockchain is approximately 17 seconds. This enables users to convert their digital assets to fiat currency in nearly an instant.

Plutus Mobile App

The Plutus Mobile App (Figure 5) connects users directly to PlutusDEX traders on the Litecoineum network. Using the Plutus internal server, the Bitcoin network can be reached using the native Bitcoin API, which is connected to via the Java API (ethereumj), or alternatively, the Javascript API (web3).

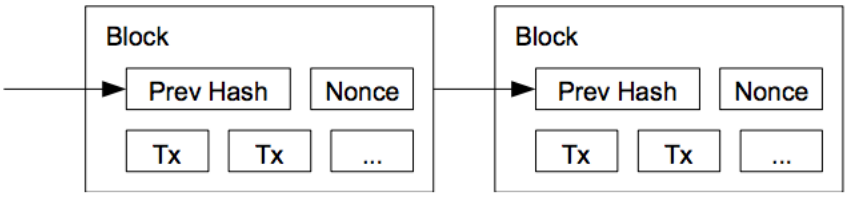
The Plutus Mobile App enables the user to choose among a selection of different fiat currencies. A virtual debit card account can be created in GBP, USD, or EUR, allowing the user to have multiple payment options.



Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with scrypt, the hash begins with a number of zero bits. The average

by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases

Network

The steps to run the network are as follows:

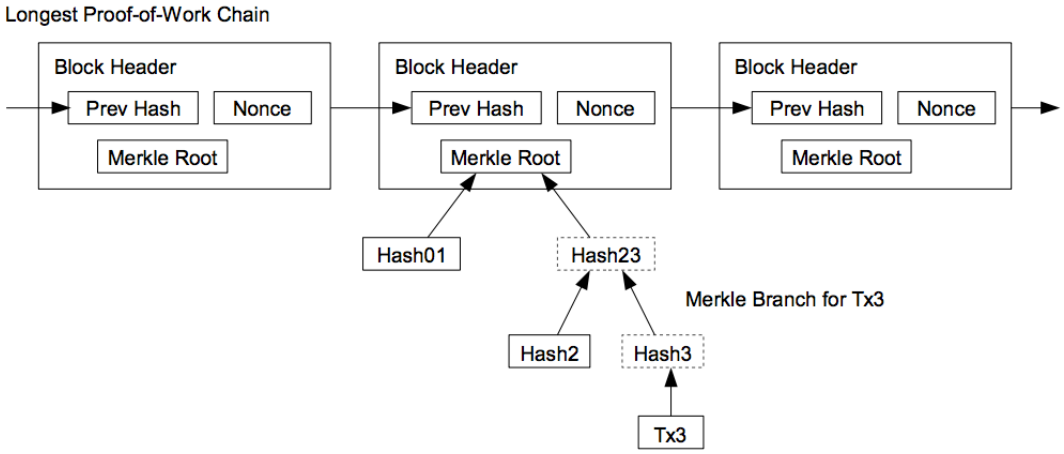
1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the node that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also

Payment Verification

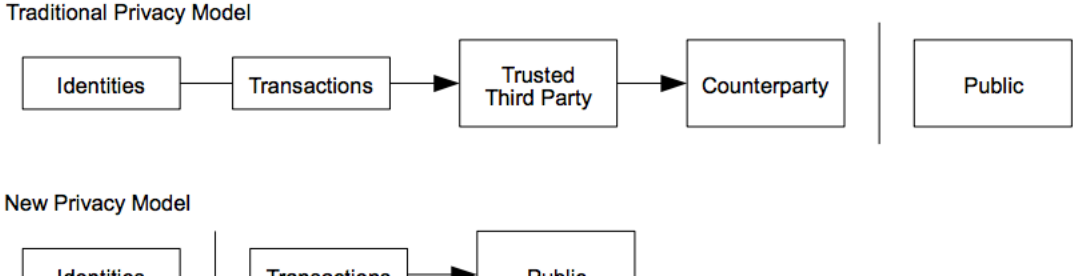
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's time stamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



Calculations

As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:
p = probability an honest node finds the next block
q = probability the attacker finds the next block
qz = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

Given our assumption that p > q, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the

$$\lambda = z\frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Incentives

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new

Glossary

Bitcoin: a digital peer to peer currency.

Blockchain: a permissionless distributed database that maintains a continuously growing list of data records secured from any tampering or revision.

Escrow: Bank account to secure fiat deposit from the network.

Litecoin: a next-generation application platform based on blockchain technology.

Fiat Currency: National currency issued by a central governing organization such as the European Bank or the Bank of England or the US Federal Reserve.

Near Field Communication (NFC): Allows a mobile phone to communicate with a payment terminal.

peer-to-peer (P2P): A network in which each computer can act as a server for the others, allowing shared access to data without the need for a central server.

Plutus Mobile Application: Allows conversion of Bitcoin and Plutuscoins to Fiat currency balance on a virtual debit card token to pay merchants.

Virtual Debit Card (VDC): A debit card that is issued and usable on a digital device without a corresponding plastic card.

Smart Contract: A computer protocol that facilitates, verifies, or enforces the negotiation or performance of an agreement between multiple parties.

Trader: Plutus app user who buys Bitcoin in exchange for for fiat currency.

User: a consumer with an NFC-enabled mobile device with the Plutus Mobile App installed on their device.

References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998. \

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

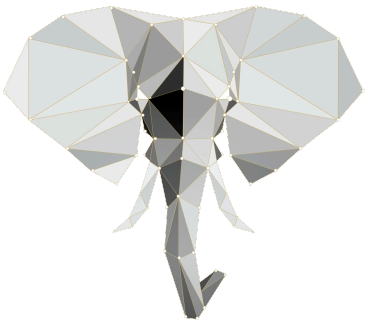
[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

Contacts

General Inquires

Hello@plutuscoins.com



Plutuscoins.com