Aryacoin : A Peer-to-Peer Electronic Cash System

sales@aryacoin.io

Aryacoin.io

Abstract : A purely peer-to-peer version of electronic cash would allow payments to be sent and received directly from one party to another , allowing parties to transfer funds across countries without any restrictions . Digital signatures prevent double spending , low transfer fees allows moving huge amounts with very less fees. Proof of work allows each transaction to be verified and confirmed . Anonymity allows users to use the coin anywhere and anytime .

1. Introduction

When bitcoin was launched it was revolutionary allowing people to transfer money to anytime and anywhere with very low transaction fees . It was decentralized and their is no third party involved in the transaction , only the sender and receiver were involved. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.
Bitcoin was made so that it would not be controlled or regulated but now exchanges and governments are regulating bitcoin and other cryptocurrencies at every step. Aryacoin was developed to overcome these restrictions on a free currency.

2. No Restrictions

The coin is a lot similar to Bitcoin and Litecoin , except the usage in the real world and how it will be traded and exchanged by the users . The user's can buy/sell the coin without restrictions and without verifications as there would be sales platforms that will be setup in locations where people can just go in and trade the coins without restriction . This enables the coin to be truly anonymous and gives it an usage in real life .

3. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After

each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

4. Timestamp Server

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

5. 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 SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work 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.

6. 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.

7. Incentive

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 coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

8) Anonymity

The coin provides decent level of anonymity for all its users. The users can send their transactions to any of the public nodes to be broadcasted , the transaction sent to the nodes should be signed by the private key of the sender address . This allows the users to use the coin anywhere any time , sending transactions directly to the node allows users from any place and country .

9) Real Life Usage -

Our team is continuously developing new and innovative ways to use the coins , we are currently developing exchanges where the users can exchange the coins without any fees and any restrictions . The coins can also be used on our other platform - mrdigicoin.io .

Along with the exchange we are currently developing other innovative technologies which would allow users to spend our coins everywhere and anywhere .

10) Offline Exchanges -

We are also working with different offline vendors which would enable us to buy and sell the coins directly to our users on a fixed/variable price this would allow easy buy/sell directly using cash . This would allow the coins to be accessible to users without any restrictions which most of the online exchanges have , also increase the value and number of users along with new ways to spend the coin. This would increase anonymity level of the coin . And introduce new users into the cryptomarket and technology . Creating a revolution which educates people about crypto and introduce them to the crypto world which introduces a whole new group of people into crypto and a move towards a Decentralized future!

11) Transactions

Transactions working is close to the transactions in Bitcoin , a transaction is divided into inputs and outputs , a transaction can at most contains 2 outputs at most - one for payment and one to return change . The number of inputs could be 1 or more . It could be one input with large value or could be many small inputs with small amounts. Transactions can be signed and sent directly to the nodes to keep anonymity and privacy . Transactions allow transferring huge amounts with very low fees and low difficulty allows faster confirmations.

12) Conclusion

Aryacoin is a coin which can be used by anyone looking to use cryptocurrency which allows them to keep their privacy even when buying/selling the coin along with while using the coin during transactions . Proof of work and cryptographic hashes allows transactions to verified . Along with using computing power for transaction verification and providing the miners a reward . Exchanges and offline vendors allowing users to use the coin without any restrictions . Cryptographic signatures allow users to send transactions directly to nodes , to allow user privacy along with keeping integrity of the data passed during the transactions.

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.