Token: A Decentralized Anonymous Messaging Board

Abstract - This paper proposes an anonymous messaging system based off the Ethereum network and the zero-cash protocol. The system takes advantage of Ethereum's blockchain and zero-knowledge proofs to guarantee that users remain completely anonymous.

Introduction/Problem

In countries and areas where free speech is limited there needs to be forum where individuals can communicate with the world without consequence. Stanford recently proposed a solution called Riposte, an anonymous, twitter-like program. Users send their message to the system, which is then XORed with a randomly created bit string, broken up, and sent to multiple secure clients. After a set amount of time the clients combine the broken up messages and publish the whole message to a main central server that holds all of these messages "anonymously" for people to view. A flaw in this system is the centralized servers that can be compromised or shut down by the government or other agencies. To solve this problem, we propose a decentralized system where the messages are instead stored on an already existing blockchain e.g. Ethereum blockchain. We create this pure anonymous system by using a form of zero-cash protocol and Ethereum mining.

2. Ethereum

Ethereum is an existing blockchain that was developed as a way for people to use a stronger language for blockchain based applications other than the current Stack Based language and physical caps that BitCoin currently has. Ethereum uses a Turing complete language that allows for stronger applications such as smart contracts to be built. This blockchain uses Ether as a form of currency or fuel, often called “gas”. Developers pay Ether to run their programs and miners get paid ether to mine blocks. Developers are encouraged to keep their programs memory efficient because the less memory used by the application, the less Ether they are required to pay. Ethereum also stores their data differently from Bitcoin, using Patricia Trees in Block Headers to store receipts, transactions, and the state. This allows for advanced applications such as smart contracts to be developed. We chose Ethereum for our program because it allows users to pay Ether in order to publish a message to the network and allows for us to store messages in a block chain efficiently to create a forum like anonymous database which can never be changed.

3. Ether Monitoring Contract/Burning of Etherium

Ethereum uses smart contracts signed by all parties participating in an agreement. This features runs agreement code after certain conditions are met, ensuring that both parties honor the agreement. We use this to guarantee that a user will receive a token once they send ether to a dead address. The token can then be

used to broadcast a completely anonymous message. While we do believe it is unnecessary to convert our token back to ether, it is possible to do so as zero-coin does, however it adds a potential danger. As the program stands now, the developers do not make money, if developers were willing to lose a little security they could have some of the Ether from the payment of the token added to an accumulator and sent to them.

This idea was based off the Zerocash protocol. Instead of being sent to an accumulator, the ether payment is simply sent to a dead address which essentially renders it inaccessible, but still shows that the currency was spent. The currency could be redeemed using an accumulator and zero-knowledge proofs, but this would result in a loss of anonymity.

4. Breakdown of Zero-Coin

The zero-cash protocol is an extension of the current bitcoin system that allows for completely anonymous transactions. Users exchange BitCoin for an anonymous currency called zero-coin. Zero-coin transactions achieve anonymity through the use of zero-knowledge proofs.

In order for an individual, Alice, to mint a zero-coin, she would first create a random serial number S. She then creates another random number r and encrypts S using r. In the case of zero-coin this is done using a Pedersen commitment, which is given as $C = g^S h^r \pmod p$, where g and h are known to all parties. Note that in order for the system to work, C must be a prime number in order for the coin C to be converted

back into bitcoin, which will be covered later on. The coin C is then added to the accumulator by miners. The accumulator is a data structure computed as $A = u^{c_1 c_2 c_3 \ldots c \ldots c_n} \pmod N$ where the integers *A*, *u* and *N* are known to everyone, and C is the same C mentioned previously, which is the Pedersen commit of S and r. At the same time, the amount of bitcoin equal in value to the denomination of the zerocoin is added to a zerocoin escrow pool.

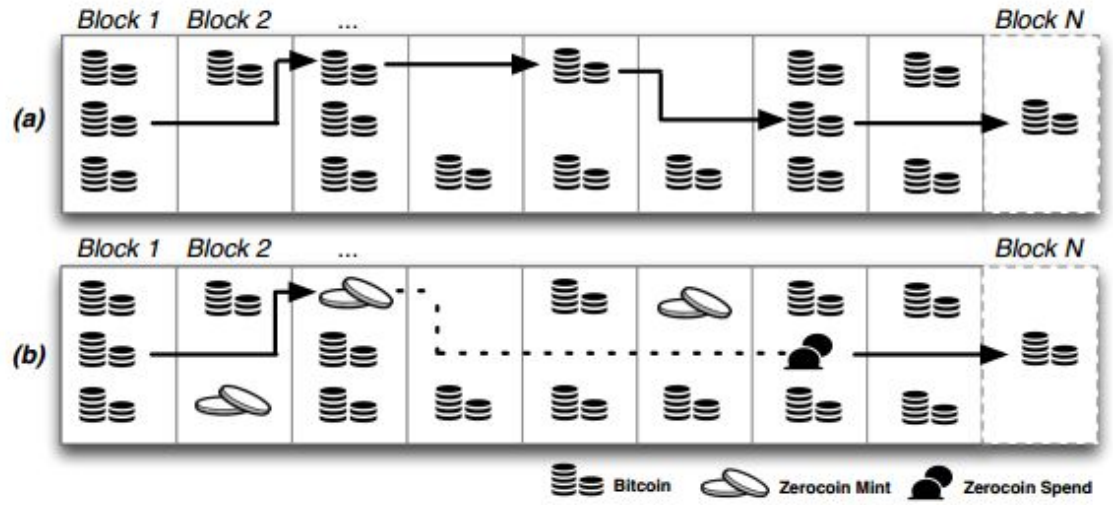

Figure 1: Two example block chains. Chain (a) illustrates a normal Bitcoin transaction history, with each transaction linked to a preceding transaction. Chain (b) illustrates a Zerocoin chain. The linkage between mint and spend (dotted line) cannot be determined from the block chain data.

To redeem the zerocoin into bitcoin (preferably to a new public address) the owner of the coin needs to prove two things by way of a zero-knowledge proof. First, Alice must prove a coin C is a minted coin in the accumulator without revealing coin C, which is done by using the accumulator. We also need the "witness," w, of a coin C, which is the value of the accumulator before C is added. The owner of the coin can verify that the coin *v* was accumulated in *A* if $w^v \bmod N = A$. Second, Alice must prove that she knows a number *r*, that along with the serial number *S* corresponds to a

zerocoin. Guessing this is computationally infeasible due to the Pedersen commit. The proof and serial number *S* are then posted as a zerocoin spend transaction. Miners then verify the proof and that the serial number *S* has not been spent previously. After verification, the transaction is posted to the blockchain, and the amount of bitcoin equal to the zerocoin denomination is transferred from the zerocoin escrow pool.

We see the escrow pool of zerocoin as a weakness in the system, as it stores all of the bitcoin used in the system at one centralized location. This is a necessary facet of zero-coin because it allows users to convert zero-coin back into bitcoin to purchase items on the network. In our system all ether converted into our currency should be used to post anonymous messages, and if any ether is wasted, it should be a negligible amount, so users will have no need to convert their tokens back to ether.

A problem that arises with this system is that the accumulator must be re computed every time a new zerocoin is minted or a zerocoin is converted back into a bitcoin, which adds time to the process to verify a transaction.

In our case users would exchange ethereum for our currency via a smart contract which provides a user with our token if they send their Ether to a dead address. By sending it to a dead address no one gains any ether, however this system is necessary in order to flooding of the network, and time and space is saved by not including an accumulator.

5. Broadcast of Message on Blockchain

Mining Ether secures the network by verifying computation and making it computationally infeasible to revise past publications. Mining Etherium uses a proof of work algorithm in which each block is attached with a nonce and mining the block is accomplished by finding the correct nonce for that particular block. To publish a message, users pay a token and append their message to a block, hash the block with a nonce and then send that block across the network all through the program's client. Once that block's nonce is found the miners blast the solution throughout the network and the block is published to the blockchain, after which the miner receives Ether for mining that block. The ethereum network uses a proof of work called Ethash which is a memory hard puzzle making it ASIC resistant. This is different from the Bitcoin puzzles which allow for ASIC farming to occur. Memory hard puzzles allow for verification with both low CPU and small memory making it more friendly for the average user to mine blocks. The memory hard puzzle requires choosing subsets of a fixed resource dependent on the nonce and block header. This resource is known as the DAG, and it changes roughly every 100 hours which prevents ASICs from being effective. Once the block is added to the blockchain it allows for an anonymous message to be seen by anyone viewing that blockchain essentially creating an anonymous forum.

6. Security

Since a given entity has to pay more and more ether the longer they choose to host their application in the ethereum network that this proposal is going to be built off of, this discourages and reduces the likelihood of attacks such as distributed denial of

service (DDOS) because it is financially unfeasible for a user or organization to flood the network with their messages. If anything, this is more likely to benefit the miners who are present in the network because a portion of the substantial amount of money that the attackers will have to spend in order to orchestrate something like this will be given to the miners who are able to successfully mine these "transactions", which therefore increases people's chances to earn money as they have a higher pool of transactions which need to be mined that they can work with. Apart from standard denial of service attacks, this will have its fair share of security protections against many other types of potential attacks or attempts to exploit the system. As mentioned above, the Ethash component which uses SHA-256 creates a puzzle that is designed to be purposely take advantage of a machine's GPUs and to be harder to be cracked using an ASIC, so that people are prevented from having an advantage over others by having multiple of these set up so that they are more likely able to mine any given transaction.

7. Conclusion

We have proposed a method to post messages online to a decentralized system completely anonymously. We start by proposing a framework using the Ethereum network and the zero-cash protocol. We showed how Ethereum's smart contracts would allow us to exchange Ether for our tokens. Once a user sends an Ether payment to a blank address they would receive an amount of our token to pay for messages to post. The message blocks go through proof of work before they are added to the blockchain

upon which an easily accessible message board is created by looking through the blocks at the messages stored, unlinkable to the user that sent them

References

https://isi.jhu.edu/~mgreen/ZerocoinOakland.pdf

https://wiki.anoncoin.net/Commitment_scheme

https://wiki.anoncoin.net/Cryptographic_accumulator