Tangram: Privacy by Design.
A Peer-to-Peer Electronic Cash System

Matthew Hellyer

July 23, 2022

Abstract

A rapid shift from traceable and non-private peer-to-peer transactions is increasingly becoming prevalent in blockchain and next generation distributed ledgers. We propose a solution that is private, non-traceable, with on-chain unlinkability, fungibility and a secure network and platform. Tangram is private by design. Transactions have no amount and addresses are stealth, fast and feeless. This provides complete fungibility.

Keywords: anonymity, applications, financial privacy, zero knowledge, network application, distributed ledger technology, cryptocurrency

Introduction

We live in a world of pervasive sensors, mobile devices, and digital services that exist to capture vast amounts of information about us every minute of our lives. Fed to an army of proprietary and often biased algorithms that we have no legal right to examine, our data might be used to determine if we are "eligible" for a job, a loan, a lease or a medical procedure. This has become the greatest accelerator of income inequality the world has ever seen, enabling the wealthy to line their pockets with untold and undeclared riches, while creating new digital divides. At the same time, hacks and leaks of our personal information that we cannot defend, are being stolen from ill-defended systems. Such attacks have become so commonplace that they are no longer causing alarm, and seldom are there consequences for the companies responsible.

> *"All human beings have three lives: public, private, and secret."*
> – Gabriel García Márquez [8]

What is needed is privacy by design, an anonymity-matters-first principle system that not only provides both, but one that is cryptographically secure and that is provable without a third party's trust. We also propose a system that is transparent in its values, one that provides proof and creates trust within the system but is trustless in its operation between users within the network.

1 Block Minting and Randomness

Block minting runs on every staking node to pre-determine the author of the new block. Each block proposal runs in 5 second slots and uses a randomness cycle to determine if the node can participate in the running slot. If there is no slot selected, the node will wait until the next 5 second slot that becomes available. Staking nodes participate in a random lottery we call the kernel, that takes place in every 5 second slot, and determines whether they are block candidates for that slot. When there are multiple staking nodes, they are participating asynchronously to become the next block candidate and mint a block and then broadcast it to the network. At that point, it's a race. The staking node who solves the block puzzles the quickest wins and there can only be one winner. If there is more than one winner, the decision is then made based on the highest staking amount.

2 The Kernel

A verifiable random function (VRF) [8] is a cryptographic operation that takes some input and generates an output along with the proof that can be cryptographically verified as random. Slots are units of time 5 seconds in length. Each slot can be a block minting candidate, but some staking nodes may be busy or have failed the kernel. In every slot, each staking node executes the VRF which then take the following inputs:

- A secret key only known to the staking node.
- Randomness values of hash inputs of the previous block, the current transactions and the block round (the block height).

The outputs are the random result and the proof that the inputs can be verified to be correct. The result (weight threshold) is then compared to the target (the proof). If the weighted threshold is less than the target, then the candidate block can move on to the rest of the protocol. Each staking node will have different outputs and will never be the same.

3 The Solution (Lightweight Proof-of-Work)

The solution takes the inputs of the result (the kernel) and the target (the proof) in which the staking node can prove to the rest of the network that a certain amount of computational work has been expended. Each solution cycle is 10 seconds in length where the target (the proof) is multiplied with the number of iterations then compared to the result (multiplied by the target int value and the kernel bit value). If the result value is less than the target value, then we have found our solution. Staking nodes that fail to find a solution will wait for the next 5 second slot. The value (uint64) of the solution will be used to calculate the network shared with the running distribution (pre-determined from the previous proof-of-stake block) and the block round (the block height). The network share halving follows the same formula that is used in bitcoin.

```
1   var halving = (int)(height / LedgerConstant.BlockHalving);
2   if (halving >= 64) return 0;
3   var sub = unchecked((long)LedgerConstant.RewardPercentage
        * LedgerConstant.Coin);
```

```
    sub >>= halving;
    return solution * (decimal)sub / LedgerConstant.Coin /
        LedgerConstant.Distribution;
```

We can then calculate the amount required by the staking node to prove a) the staking node has the required funds and b) prove to the network that the funds are valid. Verifying the solution is easily compared to proving the proof-of-work hash in bitcoin.

```
  if (LedgerConstant.SolutionThrottle > solution)
    {
    var target = new BigInteger(1,
        Hasher.Hash(vrfBytes).HexToByte());
    var weight =
        BigInteger.ValueOf(Convert.ToInt64(solution));
    var hashTarget = new BigInteger(1, kernel);
    var weightedTarget = target.Multiply(weight);
    isSolution = hashTarget.CompareTo(weightedTarget) <= 0;
    }
```

4 The SLOTH (Verifiable Delay Functions) [3]

We use the slow-timed hash function that take inputs from the VRF signature and the staking amount and return a nonce. The inputs are used in such a way that it's impossible to choose these parameters but possible to compute the sloth that takes wall-clock time at least the same amount of time in seconds, regardless of the computers available processing resource or in parallel. The time it takes to verify the nonce (which is the correct output hash), takes at least the same time as a wall-clock is required to expend the same number of computations (in seconds). But the output proof is quickly verifiable. The sloth is there to impose a time delay before the block can be broadcast to the network. This prevents malicious actors from influencing the block minting, since all inputs are finalised before the sloth is finished computing.

5 Fork Choice Rule [2]

The objective in any ledger is to reach a consensus on which blocks to discard or to maintain. In the Nakamoto protocol it's the longest chain rule. In proof-of-stake systems this can lead to serious security issues. Our proposed protocol is not subjected to bribery attacks, or a strategy known as nothing-at-stake. Grinding attacks are prevented as there are no elected leaders and the stake amount is not known beforehand. A corrupt node can always wait until some time slot and propose a new block, it cannot include transactions that don't exist at any depth. Nor can the corrupted node change the staking amount that hasn't been defined by the solution or the network share.

When considering a corrupt node grinding its own private chain at any block, the corrupt node grows the chain without broadcasting to the network until it is certain that it can overtake the public ledger. Note, that the public ledger grows the chain at the rate proportional to t. The private chain also grows at the rate proportional to

t. Such an attack is known as coin grinding, and it can alter any content within the private ledger and sign all blocks again. To prevent this private coin grinding attack, we are using an s-truncation fork rule.

An honest node safeguards by choosing the right fork rule in the following protocol. Let bfork be the block where the two chains fork. The honest node compares up to s blocks after the fork. The chain with the shortest time for those s blocks is chosen. If the honest node's chain is longer in time, then we compare which chain for those s blocks have highest staking amount. The chain with the highest staking amount at that point has expended more time using the solution and the VDF. The chain is verified and accepted if those s blocks can be attached before the fork and the transactions exist in blocks with depth L.

6 Unlinkability Transactions

Understanding privacy and anonymity implications (ie: linkability) will be helpful for users interested in maintaining maximum control over the visibility of their transaction(s) and storing of their coins. Transactions like Bitcoin and other non-private ledgers where Alice sends to Bob, the direct linkability with the coins transfer is broken. T. Okamoto K. Ohta describes privacy: "relationship between the user and his purchases must be untraceable by anyone" [7, 9]

Two properties which a fully anonymous electronic cash model must satisfy [1]:

1. Untraceability: for each incoming payment all senders are equal.
2. Unlinkability: payments are concealed from third party's view as its impossible to prove they were sent to the same person.

One-time addresses (payments) are an anonymity technique that protects the privacy of the receiver while preventing observers from knowing your transaction history [1].

6.1 Coinstake Unlinkable Transaction

These transactions are the main aspects within the proof-of-stake consensus. The incentive to running a node is for those who are more privacy conscious. Although the node commits a coinstake (stake amount) and coinbase (reward) transaction, it never reveals anything about the incoming and outgoing payments. These are unlinkable and untraceable.

The proof-of-stake transaction:

1. Coinbase: reward amount in the clear, locked for 24 hours
2. Coinstake: proof of amount in the clear and spendable
3. Change: balance is hidden (Pedersen commitment) and spendable

Each staking node runs in 5 second time slots, when the node is selected for that time slot the node initialises the staking wallet which is built into all nodes. The staking node passes the inputs, stake amount (bits), network share (reward) and the sender's address. When the wallet successfully completes the payment, it is then returned to the block proposal. Upon completing the rest of the protocol, the staking node broadcasts the block proposal to the network where it will be verified or rejected.

The amounts in the clear will not break unlinkability or untraceablity on either coinstake or coinbase transaction and are openly displayed. This is because a) honest

staking nodes have followed the protocol correctly and b) have committed to the required amounts that are verifiable.

6.2 Standard Unlinkable Transaction

All transactions are equal except for the coinstake and coinbase transaction where the amounts are in the clear. Each sender transfers payments to the next by using ring signatures to digitally sign every payment. Ring signatures are defined as "1-out-of-n signature convinces a verifier that a document is signed by one of n possible independent signers without allowing the verifier to identify which signer it was" [7]. Our protocol is base on MLSAG Ring Confidential Transactions [6]. To conceal the amounts the basic idea is based on Confidential Transactions [5].

There are no fees when sending a transaction but it requires some computational work before the transaction is broadcasted to the network. As there are no fees this makes nodes vulnerable to flooding and DoS attacks. We need a way to slow down such attacks. For our purposes, we use timestamping and a VDF (verifiable delay functions) to prove that some time has been expended. The trade-off to this approach is the wait time before the transaction is broadcast to the network. However, the advantage is we can use this approach as a substitute for paying in time and not in fees. Higher wait times on the transaction placed in the pool with other pending unverified transactions will get priority over transactions with lower wait times.

7 pBFT (Leaderless Practical Byzantine Fault Tolerant)

We use Blockmania [4] which is a leaderless pBFT consensus. Distributed consensus protocols (algorithms) provide the necessary means for a distributed network or system to agree, verify and validate transactions without the need for a central authority.

Most pBFT-based consensus algorithms follow the design of having a constant leader who is a primary node and who accepts all requests while proposing blocks to the rest of the network from other nodes. This means that these networks are synchronous and may cause further challenges such as network communication errors. In addition, a malicious leader may cause hindrance to the network.

In a leaderless based pBFT system, leaders are executed on a first come first see block basis, i.e., when a block gets seen in the network it will start a round with each node in its membership ring, and this continues with each new block, allowing for asynchronous communication. Anyone operating through a light-wallet and submitting transactions may connect to any node and submit their transactions. They may then observe and determine if their transaction was successful, and if for any reason it was not, they may select a different node and re-transmit the transaction.

8 Network

The steps to the network are as follows:

1. A single transaction is broadcast to a seed node which a wallet operator can choose.
2. New transactions are broadcast from the pool to all nodes immediately.

3. Each node collects and verifies new transactions before the transactions are placed into the proof-of-stake pool.
4. When a node is selected for the 5 second time slot (the kernel), the node verifies the owners did not double-spend the coin. Unverified or double-spent transactions will be discarded.
5. When the node finds the solution (lightweight proof-of-work), it can then calculate the network share (coinbase). The coinstake (bits) = (solution * network share / 8192).
6. The node executes the VDF and can compute $(y, \pi) \leftarrow \text{Eval}(pp, x)$ in t sequential steps, then it broadcasts the block to all nodes.
7. Block broadcasting happens in rounds. When the round (block height) has been seen by the network, the node accepts the block, only if all transactions in it are valid and not already spent.
8. In next running time slot, if the block was accepted the node starts working on the next block.

Nodes consider the fork choice rule to be the correct one and will keep on growing the chain. Blocks are broadcast in rounds (block height). If two nodes broadcast different rounds and the first node is ahead in rounds, it will broadcast the seen block round back to the second node but will discard the block. Once the node has received the correct round and all transactions are valid, it continues to the next block.

There are two node setups. Both are identical except for staking. A node operator can choose either the relay node or the staking node.

1. Relay: A participating trusting node that observes traffic, validates blocks, transactions and keeps a full ledger.
2. A full trusting node that participates in block minting and a relay node.

Relay and staking nodes run in the same manner accepting rounds in sequential order, some nodes will receive rounds in the correct order, other nodes might fall behind or could grow the chains with other nodes that are on the same round. We can say that these nodes have forked. In this case they work on the round they received, but discard rounds that are either ahead or behind. Transactions are valid in one or more clusters of nodes where the transactions do exist and are not spent. Forked nodes that are ahead or behind, receiving new transactions (that do exist and is not spent), will accept these transactions on the forked chain. There are no differences between transactions on nodes ahead or behind in rounds where transactions can be verified, and the owners did not double-spend their coins. At some point these nodes will come together and exchange block information and follow the fork choice rule. Which is where a node will decide whether to maintain or discard the blocks where the two chains have forked.

9 Verification Rules

For a block to be considered valid, it must have followed the rules:

1. The sloth finds a y for which $\text{Verify}(pp, x, y, \pi) = \text{Yes}$, but $y \neq \text{Eval}(pp, x)$.
2. Coinstake transaction amounts are verifiable, that the network share and solution meet the required previous (block) distribution calculation.

3. The kernel was selected for the round (block height) and Verify$(pp, x) =$ Yes.
4. The kernel proof output is correct for $Verify(pp, x, y) =$ Yes, where pp is the node public key, x is the VRF proof and y is the kernel.
5. The solution is verifiable for Verify$(x, k, s) =$ Yes, where x is the kernel proof, k is the kernel and s is the solution.
6. The bits (reward) for Verify$(x, y) =$ Yes, where x is the solution and y is the network share.
7. Block header lock-time is unlocked.
8. Block header merkle verifies the previous block merkle and the current hashed transactions.
9. Owners did not double-spend the coins.

For a transaction to be considered valid, it must have followed the rules:

1. Transaction attributes are correct.
2. A payment-only transaction used the sloth and finds a y for which Verify$(pp, x, y, \pi) =$ Yes, but $y \neq$ Eval(pp, x).
3. The spendable commitment exists, and the ring signature members commitments exist.
4. The key image does not exist.
5. A coinbase only transaction lock-time is unlocked.
6. A coinbase only transaction where Commit$(r, y) = amount$, where r is the reward and y is the blind.
7. A coinstake only transaction where Commit$(s, y) = amount$, where s is the stake and y is the blind.
8. The total commitment $sum = 0$.
9. The bullet proof can Verify$(x, y) \neq -1$, where x commitment and y is the proof challenge.
10. MLSAG computes the values L', R' and the hashes $c'1$ to $c'n$. If $c1 == cn$ the signature is accepted.

10 Emission

We use the term network share as the emission process for the overall amount of Tangram digital coins. To ensure the network share (emission) process we use the following formula for block coinbase transactions: Number of blocks before the halving $25246081 \cdot 0.08333333/60/365.25 = 3.9999999984$

BaseHalving = height$/25246081$

BaseSubsidy $= 50 \cdot 10^9$

BaseHalving >>= BaseSubsidy

NetworkShare = solution $\cdot$ (BaseSubsidy$/10^9$/MSupply)

11 Conclusion

We propose a system which includes its networking capabilities, consensus method for important transparency without the reliance on trust needed in a privacy based system.

References

[1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n Signatures from a Variety of Keys. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–432. Springer, 2002.

[2] Vivek Bagaria, Amir Dembo, Sreeram Kannan, Sewoong Oh, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Proof-of-Stake Longest Chain Protocols: Security vs Predictability. *arXiv preprint arXiv:1910.02218*, 2019.

[3] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[4] George Danezis and David Hrycyszyn. Blockmania: From Block DAGs to Consensus. *arXiv preprint arXiv:1809.01620*, 2018.

[5] Greg Maxwell. Confidential Transactions. Technical report, 2015.

[6] Shen Noether, Adam Mackenzie, et al. Ring Confidential Transactions. *Ledger*, 1:1–18, 2016.

[7] Tatsuaki Okamoto and Kazuo Ohta. Universal Electronic Cash. In *Annual international cryptology conference*, pages 324–337. Springer, 1991.

[8] Trevor Perrin. The XEdDSA and VXEdDSA Signature Schemes. *Specification. Oct*, 2016.

[9] Nicolas Van Saberhagen. CryptoNote v 2.0. 2013.