Merit: Digital Money Built for People

Maxim Khailo
Adil Wali

August 21, 2018

Abstract

A safe and fair digital money would allow all users to participate and benefit from the economic freedom that blockchain technology can provide. Yet, existing blockchain systems function as bearer currencies and struggle to support everyday needs of most users. Centralization has overrun these systems increasing the reliance on trusted third-parties. Additionally, the technological limitations of these currencies allow for massive fraud and theft and create an unreasonable burden for the average user to both stay safe and conquer complex usability barriers. Merit attempts to address these issues with structures that encourage growth and ease of use while allowing users to keep their assets safe in decentralized vaults. To combat centralization with mining, Merit implements an ASIC resistant Proof-of-Work based on the Cuckoo Cycle. It further innovates by introducing Proof-of-Growth, which enables ambassador mining rewards that reward users for growing the network of Merit users.

1 Introduction

Nation states designed their banking systems to facilitate their needs. A nation state needs to be provisioned and uses the banking system to purchase goods and collect taxes. Nations also desire to keep the populace happy by facilitating trade and everyday transactions in market systems. Cryptocurrencies have developed to subvert the needs of the state and elevate the needs of the people as the primary goal. All cryptocurrencies focused on undermining the state do so at the expense of the primary intended purpose of giving financial freedom to everyone. Many cryptocurrencies have ended up with extreme centralization in their operations, both in mining and in facilitating exchange. People, desperate for usability, are relying on third-party providers to bridge the gap between the protocol and the user. The outcome has become an anathema to the original vision of decentralized cryptocurrencies. Merit believes this to be the single largest problem facing the cryptocurrency landscape today. The Merit Foundation has set out to address this problem on three fronts: simplicity, safety, and community. Merit's high-level vision is to finally attain the original vision of a decentralized internet-of-money where individuals don't have to rely on third-party trust to benefit from the power of the blockchain.

2 Themes

Merit has aggressively focused innovating four interrelated ideas.

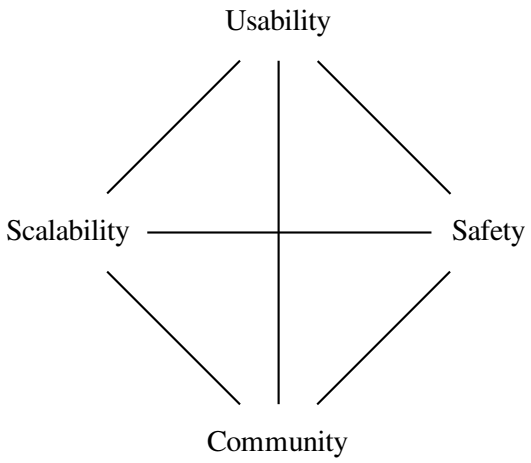

Figure 1: The Four Themes

To deliver on the community theme, Merit is implemented as an invite-only currency that creates a scarcity of addresses. The invite-only mechanic enables stable growth, improved safety, and innovative features to develop a platform for building communities.

To deliver on safety, Merit employs decentralized vaults which provide whitelists, rate limiting, and decentralized account recovery. We can think of vaults as the mssing savings account on the blockchain.

To deliver on usability, Merit provides an identity system which provides user-friendly aliases such as @max or @adil in liue of addresses. Merit also provides a way to send transactions through any communication channel via a secure URI schema.

To deliver on scalability, Merit's first implementation has 80 times the maximum throughput of bitcoin, through improvements in block size and frequency. Further, Merit employs a new ASIC resistant Proof-of-Work algorithm called the Cuckoo Cycle so that everyone using conventional hardware can participate. Perhaps most dramatically, Merit employs a generalization of the blockchain called the block fabric which can provide performance that is many orders-of-

magnitude higher than others cryptocurrencies.

3 The Problem

3.1 The Third-Party Paradox

The design of many Cryptocurrencies requires users to manage and secure hundreds of cryptographic keys themselves. Each transaction in Bitcoin creates a new key which the user must manage and secure. In this way, each cryptocurrency acts as a bearer currency and does not have any of the modern conveniences centralized banking systems provide. In a state run fiat-based system, if one were to lose access to their bank account, they can get that access back by simply providing proof-of-identity. If they go on to lose access to their Bitcoin keys, nobody can recover the funds. Because of how impractically burdensome this is for most users, many users of many popular cryptocurrencies such as Bitcoin, Ethereum, and Litecoin rely on third-party services to securely store these cryptographic keys. Simply stated, most cryptocurrencies avoid the realities of human nature and the human condition. As a result, counter to the core ethos of cryptocurrencies, third-party companies have have become the defacto standard for so many users worldwide.

Users of cryptocurrencies also rely on third-parties to maintain the blockchain ledgers because the vast majority of mining and node operations have become centralized. Because of the poor quality and usability of many blockchain projects, people have a difficult time using these systems in the decentralized way they were intented to be used.

Mining is also dramatically centralized. This is primarily because the Proof-of-Work approach to mining rewards those with extraordinarily large bank accounts. Companies and individuals who can fund large mining data centers full of specialized hardware dominate the hashpower of most major currencies. [1] Secondarily, the poor quality and high-opacity of much of the mining software causes users to have a difficult time in truly decentralizing these systems the way they were intended to be. Even those users who are brave enough to mine cryptocurrencies typically rely on centralized mining pools because most systems have become impractical to mine in a decentralized way.

The paradox exists because of the decentralized nature of these systems. Decentralized systems are hard to make user-friendly. This usability gap gets filled by centralized third-parties, and many non-expert users drawn to the space do not realize that this centralization is completely counter to the purpose of cryptocurrencies. Perhaps worst of all, this centralization creates a false sense of security for users who are drawn to the 'cryptographically secure' headlines. Billions of dollars have already been stolen from these centralized banks that attempt to bridge the usability gap. Merit aims to solve this usability gap so that users can get the level of simplicity that they expect without having to compromise on the core notion of decentralization.

3.2 Safety First

Merit recognized the need to aggressively focus on ease of use and safety so that users are not tempted to rely on centralized third-parties for everyday use. To this end, Merit innovates by adding the new Global Send protocal, which empowers users to send funds quickly and easily across any communication channel like SMS and email. This approach stays true to decentralization through the creation of an escrow address on the blockchain that can subsequently be claimed by the recipient. Merit also provides a way to secure funds in vaults that have whitelists, rate limiting, and decentralized account recovery. These features make theft more difficult and diminish the consequences.

3.3 The Death Star Networks

The dream of everyone running mining software has been subverted by those with the wealth to develop specialized mining hardware and use them on a large scale. To participate in mining for Bitcoin and other popular currencies now requires investments in equipment that is out of reach for most people. Specialized hardware has led to massive centralization in the form of mining farms and pools.

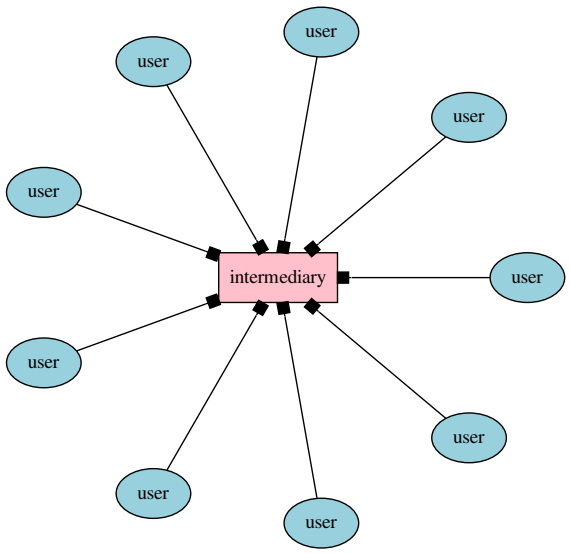

Figure 2: A Start Network

This centralization mirrors that of what happened to the internet at the end

of the dot-com bubble. We ended up with massive star-shaped networks where intermediation has become the norm.

Merit combats this centralization by using a memory-hard ASIC resistant proof-of-work algorithm known as the Cuckoo Cycle [2], which relies on the commodification of RAM to resist any advantage specialized hardware would have.

4 Community

The Merit team believes building a healthy and active community is one of the best ways to prevent centralization. We believe that incentives largely shape the behavior in human systems, and that the incentives in the cryptocurrency world are woefully incomplete. Before Merit, Proof-of-Work mining has primarily been a security activity. The newer notion of Proof-of-Stake rewards 'buying and holding' behavior that indirectly takes currency out of active daily use in the system. Neither of these incentives addresses the most important pillar in any currency: the number of people using it. Merit incentivizes the creation of this lifeblood through its innovative Proof-of-Growth algorithm that drives ambassador mining. We recognize that not all users are cut from the same socioeconomic cloth. Not everyone is wealthy enough to build or buy a datacenter or to buy millions of dollars worth of a crypto and hold it. But anyone, irrespective of wealth or background, can share and grow the community. We call this crucial user the ambassador, and we rely on them to provide the valuable function of educating users and providing a voice and perspective of what cryptocurrency and Merit are all about.

4.1 Proof-of-Growth

To use Merit, you must beacon your public keys by providing an address of an inviter. The address of a public key is the same as that of Bitcoin and other cryptocurrencies. In this case, it is the public key hashed using RIPEMD-160 with base-58 encoding [5].

Through the activities of the invitee, the inviters address becomes a candidate in a lottery that is executed when every block is mined. The miners dedicate a percentage of the mining reward to users called Ambassadors of Merit. This beaconing system constructs an Ambassador Forest.

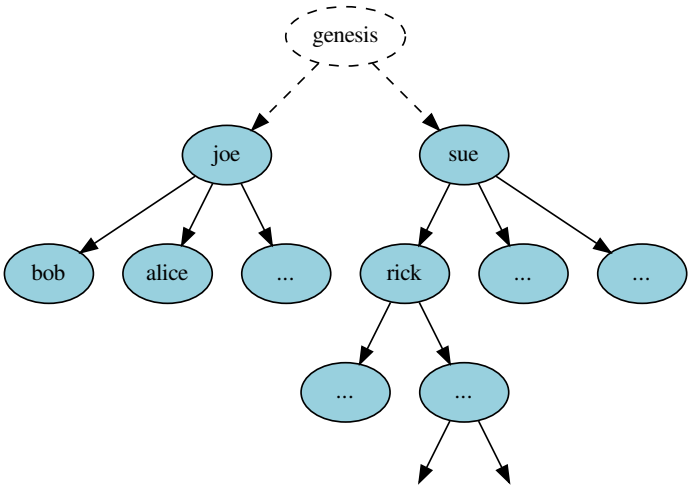

Figure 3: Ambassador Forest

It is a forest because the genesis address does not participate in the lottery system. Each node in the Ambassador Forest has a Community Growth Score (CGS) which is computed as a function of the node, and its children's value in Merit. The higher the CGS, the more likely the Ambassador will win a reward in a block. The function used to compute the CGS for an address is fair, and new members can compete with older members as readily.

This beaconing process is a pre-requisite requirement for enabling a distributed invite-only system and other relevant safety and community features.

4.2 Beacons

Each block provides a dedicated space for Beacons. A Beacon advertises an address and some related data about an address. A Beacon has some required data, is signed and broadcast to the mining nodes. The mining nodes will validate the Beacon and then include it in a block.

4.2.1 Beacon Properties

The beacon is composed of...

Property	Description
Inviter Address	The address you were invited by.
Address	The address you want to beacon.
Public Key	The public key which to sign the beacon.
Signature	The signature produced by the public key.
Alias	An optional 20-byte field that can be used to assign an alias to an address.

4.2.2 Address

If the address type is a regular public key, then the address of the public key must match the provided address.

Merit inherits and extends Bitcoin's scripting system and therefore has addresses for scripts just like it does for public keys. If the address is a script, then the public key can be any public key. This key is then combined with the script' address to provide a new address which the user will advertise to others. This process is required to prevent bad actors from reassigning the inviter address to those they desire.

4.2.3 Inviter Address

Each Beacon must include an inviter address. The inviter address is required to maintain the Ambassador Forest so that the Ambassador Reward structure remains consistent. Because Merit is an invite-only system, each new address must, therefore, have an inviter address. The Inviter address must have already been beaconed and should have received at least one Merit Invite (MRTi).

4.2.4 Public Key

If the address beaconed is generated by a public key, then the provided public key's address must match the beaconed address. If the address is derived from a script, then the public key can be any key.

4.2.5 Signature

The signature is signed by the private key that corresponds to the public key. The data used to generate the signature is the serialized Beacon sans the signature.

4.2.6 Aliases

The alias in the beacon must be globally unique. An alias can be used in place of an address in all the core activities on the Merit network. Aliases provide a much easier way of identifying recipients of transactions. This alias is optional, and users can simply choose to only be known by their more classic Merit address. We sense that aliases will not only be used by most users, but will also be used

by institution to make it easy to transact with customers, donors, and payees via the Merit blockchain.

4.2.7 Beacon Lifetime

The beacon is not mined in a block until its address is invited via an invite transaction. The beacon remains in the pool until it an invite is sent to it's address, where a miner will package both in the same block. This requirement exists to prevent parking aliases in the blockchain without being invited and is an anti-spam measure for beacons.

A user can deactivate a beacon by spending the beacon's last invite. If a beacon's final invite is spent, then the alias used in that beacon becomes available again for someone else to take.

4.3 Address Aliases

A user can set an optional globally unique alias to their address via a beacon. All interfaces that require an address can use the alias instead. The alias is first come, first serve and must be globally unique. The alias is limited to 20 bytes and has some basic validation to make sure it does not use specific reserved words.

4.3.1 Simplicity

@foobar = MEegc9eva9moRxkRi74GSL7AtPVdbyCXe2

Aliases provide a great improvement in usability over existing cryptocurrencies where users can use recognizable names for their addresses. This feature is available but not required for people to use. Not all users want to have a recognizable alias of their addresses and would prefer to stay anonymous. Additionally, for public organizations, this provides a great opportunity to make receiving funds easier in the same way that DNS makes setting up internet addresses easier via top-level domains.

4.3.2 Better Safety

Aliases also provides a degree of safety because one can associate an alias with an address and can use either one to validate the other. Further, aliases are dramatically easier to type-in and spot-check than a 34-character key. If the user prefers to use an address, they can query the blockchain for the alias and use that to check that the address is correct. Organizations can publish their address along with the alias and users can look up the address for an alias and validate that the alias is correct. In these ways, aliases should help eliminate a frequent source of error and fraud when trying to transact with users who use this feature.

The Merit core team considered using Internationalized Domain Names for Applications (IDNA) [6], notably the "stringprep" [7] algorithm for sanitizing

and indexing Aliases but eventually decided against it for safety reasons. Mainly spoofing names using a homograph attack to trick users into thinking they are using one alias when in fact they are using another [8]. These attacks use combinations of Unicode characters that look visually identical to other characters. Homogrpaphy attacks are particularly dangerous if users copy and paste these aliases instead of typing them out. Modern web browsers are frequently being patched to prevent phishing attacks [9], but it is a never-ending battle. Since Merit is a blockchain of assets, the risks for phishing would be too high if aliases used Unicode. After serious consideration, it was determined that aliases should be heavily restricted at the cost of flexibility and internationalization.

Aliases must be matched the following case-insensitive regex.

Listing 1: Alias Validation Regex

```
[a-z0-9]([a-z0-9_-]){1,18}[a-z0-9]
```

4.3.3 Organized Distribution of Names

The invite-only design is also important for individuals and organizations which a unique identity for an address is important. It provides them a chance to claim an important name by seeking out an invite before others can claim it. It also provides an incentive for users to get invites as they limit access to this identity system.

4.4 Invites

Merit is an invite-only cryptocurrency where exiting users control the growth. The invite-only system is designed to provide stable growth and empower the ecosystem to grow organically. This ensures that the software can be improved in lock-step with the growth of community, and helps the whole community avoid the 'not ready for primetime' paradox that plagues the fastest-growing currencies. Further, it ensures that the network can be as secure as possible at each checkpoint of scale. The invitations are mined just like the currency and distributed in a decentralized way. The distribution of invitations is uniform among existing addresses. The creators of Merit cannot generate invites, and do not have any special privileges related to their distribution.

4.4.1 Address Validation

An address is valid when it has both a valid beacon and at least one unspent invite, known as the activating invite. As a user accrues invitations, they can transfer them to other beaconed addresses which activates those addresses on the Merit network. A user who wants to be invited signals this by beaconing their address and publicly stating which address or alias invited them. The user who owns the inviter address will be notified when the beaconing happens, and they can consider the new address for an invitation. To activate that new beaconed address, they send an invite to that address.

As long as an address has one unspent invitation, it is activated, and can thus participate in sending and receiving Merit.

4.4.2 Transactable

You can transact with invites the same as you can with Merit. You can send them to addresses and receive them. The same requirements to spend Merit are checked for invites. Invites have additional validation and you cannot mix them in the same transaction as with Merit. The inputs of invites must also be invites, and the outputs must also be invites. You can think of invites as a currency within the system which enables the use of Merit.

4.5 Ambassador Rewards

Merit recognizes that a variety of users contribute to the growth and stability of the system but might not run mining nodes. This could be for a variety of reasons, with perhaps the most compelling being that the majority of the world's population does not own a PC. Unlike bitcoin and other cryptocurrencies, Merit does not presume a homogeneous and tech-focused user base. The invite-only system requires real people to decide and distribute invites to grow the Merit community. The lottery system was designed to reward these stewards. Miners will distribute out "Ambassador Rewards" in the form of a lottery based on how well they grow the Merit user base. The rewards are distributed per block, with part of each block's total reward being split between the security miner and the top ambassador miners of each period. A deterministic lottery system executes a weighted random sampling of the existing keys and distributes proportional rewards to the winners.

As of this writing, about 10 Merit (MRT) are awarded to the miners and 10 MRT are rewarded to Ambassadors. Twenty Ambassadors win rewards per block where 5 MRT out of the 10 MRT are distributed to Ambassadors based on what we call the CGS distribution and the other 5 MRT are selected from the $SubCGS$ distribution. These two distributions will be explained in detail below.

4.5.1 Community Growth Scores

Each user in the Merit network has a Community Growth Score (CGS) which is used to decide how to give out the Ambassador Rewards by the Merit network. The CGS factors in both the balance of the user and also the balances of their network. In order to compute the CGS of an address, we must compute several different things and bring them together.

1. Aged Balances of all the addresses.
2. Address Network Contribution of all the addresses.
3. Subtree Contribution of all the Subtrees in the Ambassador Forest.

4. The Weighted Score of all the addresses.

5. The Expected Value of each address.

6. And finally the Community Growth Score of each address.

The CGS computation is a variation of the Pachira Lottree [24] which is a lottery tree that prevents sybil attacks while indirectly incentivizing network growth. In addition to indirectly incentizing network growth, we provide direct incentives by computing $SubCGS$ and performing a second lottery.

4.5.2 Aged Balance

The first step to compute the CGS is to compute the $AgedBalance$ of an Address in the network. This is done by taking all the unspent coins, and adding up all the balances applying a $BalanceAppreciation$ function to them.

$$BalanceAppreciation(h,t,m) = (1 - \frac{1}{(\frac{t-h}{\frac{m}{4}})^2 + 1.0})$$

Where h is the block height of the coin, t is the height of the block being validated, m is the number of blocks to reach maturity. Merit uses a 30 day maturity which at the time of a minute between blocks means $m = 43200$.

$$AgedBalance(b,h,t,m) = b * BalanceAppreciation(h,t,m)$$

Where b is the balance of the unspent coin.

4.5.3 Address Network Contribution

Each address in the network has a network contribution which is the sum of all the unspent coins multiplied by a $BeaconDecay$ function plus the sum of all aged balances. The $BeaconDecay$ function follows a similar curve to the $BalanceAppreciation$ except in reverse.

$$BeaconDecay(h,t,d) = \frac{1}{(\frac{t-h}{\frac{d}{4}})^2 + 1.0}$$

Where h is the height of the beacon, t is the height of the block we are validating, and d is the decay time. The decay time is 30 days of block time which means $d = 43200$.

The Adress Network Contribution (ANC) can then be computed as.

$$BC(C,n,t,d) = BeaconDecay(h,t,d) \sum_{i=0}^{n} C_i$$

$$AB(C,t,m) = \sum_{i=0}^{C_n} AgedBalance(C_i, C_h^i, t, m)$$

$$NC(C, h, t, d, m) = BC(C, n, t, d) + AB(C, t, m)$$

Where C is the set of all unspent coin balances, H is the set of all unspent coin block heights, n is the total unspent coins, h is the beacon height, t is the height of the current validation height, d is the number of blocks for decay, and m is the number of blocks until coin maturity.

4.5.4 Subtree Contribution

Each address is either a leaf node in the Ambassador Forest or a subtree. If the node is a leaf, then the subtree contribution of the address is the same as the address's network contribution specified above. Otherwise the address is a subtree and the contribution is simply the summation of all child subtree contributions.

$$SC(N, t, d, m) = NC(N_u, N_h, t, d, m) + \sum_{i=0}^{N_c^t} SC(N_c^i, t, d, m)$$

Where N is the address and it's properties, t is the currently validating block height, d is the decay time, and m is the maturity time. The address property N_u is the address's unspent coins, N_h is the addresse's beacon block height, N_c^t is the total children, and N_c^i is a child at index i. The Subtree Contribution can be efficiently computed by doing a post-order traversal of the Ambassador Forest.

4.5.5 Weighted Score

After the Subtree Contribution a weighted score is computed for each address. The weighted score is a convex function where the curve is controlled by two parameters called the B and the S parameters.

$$WS(c, T, B, S) = B\frac{c}{T} + (1 - B)\frac{c}{T}^{1+S}$$

Where c is the subtree contribution, T is the subtree contribution of the genesis address, which is the same as the entire contribute of the Ambassador Forest. The B and S parameters control the balance between rewarding growth and fairness. Both B and S must be between 0 and 1. Merit uses $B = 0.2$ and $S = 0.05$ which was determined via simulation.

4.5.6 Expected Value

The final component of computing CGS is calculating the expected value of an Address, which is the probability of the address being selected during the lottery. The expected value of an address is the weighted score of the address in question, minus the sum of the weighted scores of all addresses's children.

$$ExpectedValue(c,T) = WS(c,T,0.2,0.05) - \sum_{i=0}^{c_n} WS(c_i,T,0.2,0.05)$$

Where c is the subtree contribution of an address, c_i is the subtree contribution of the addresses child at index i, and T is the subtree contribution of the genesis address. Because WS is a convex function, the difference between the $WeightedScore$ of a node and it's children get's bigger as the network get's bigger. In other words, the WS of an address increases as the network grows under it relative to an address with the same direct contribution but without the network.

4.5.7 Final CGS and SubCGS

Once all the expected values are computed for each address, the final CGS is simply the contribution of the whole Ambassador Forrest times the expected value.

$$CGS(C,T) = T * ExpectedValue(c,T)$$

And the $SubCGS$ is computed exactly like CGS except log is applied to all the contributions.

4.5.8 Two Lotteries Per Block

Coins rewarded in merit are split three ways. For every block, half the coins go to the miner. The other half is distributed across two lotteries, one based on CGS and the other based on $SubCGS$. Each lottery is executed by applying an inverse transform sampling on the set of all valid candidates.

4.5.9 Valid Candidates

Only addresses with certain criteria are able to participate in the lotteries. An address must have the following properties.

1. Must have a balance of at least one block reward (20MRT in the beginning).
2. Beacon must be valid, meaning it must have at least one invite.

4.5.10 CGS Lottery using Inverse Transform Sampling

Each miner executes a lottery for a particular block by taking all the valid candidates and performing an inverse transform sampling [11].

The cumulative distribution function used in the sampling is computed by taking all the $CGSs$, sorting them in ascending order and stacking them on top of one another.

$$CDF(CGS_i) = \frac{PreviousCGS(CGS_i) + CGS_i}{TotalCGS}$$

To pick the 10 winners. the block hash is used as a seed value which is applied iteratively per winner to get the next winner.

$$hash = sha256(previousHash, previousWinner)$$

$$rand = siphash(hash)$$

Where $rand$ is used to search for a winner in the CDF space. This search in $O(logn)$ time by doing a lower bound search given the rand value in the CDF space. The overall complexity of the sampling is $O(nlog(n) + slog(n))$ where n is the number of candidates and s is the number of samples.

4.6 Invite Rewards

The algorithm for creating and distributing invites is similar to the Ambassador Lottery except there are three lotteries.

1. The Lottery based on $SubCGS$.
2. The one-time award lottery.
3. The random lottery.

Each of these lotteries has a specific purpose to improve the overall experience of old and new users.

4.6.1 Selecting Addresses

As the blocks progress, Merit needs to decide whether to distribute invites or not. At a minimum one out of ten blocks, the miner is awarded an invite and one out of ten blocks, one invite is distributed to an address based on the three potential lotteries. Note this is the minimum amount of invites distributed and Merit will adjust the amount upwards as more invites are used by doing a moving average over the last X blocks. Typically over a days worth of blocks.

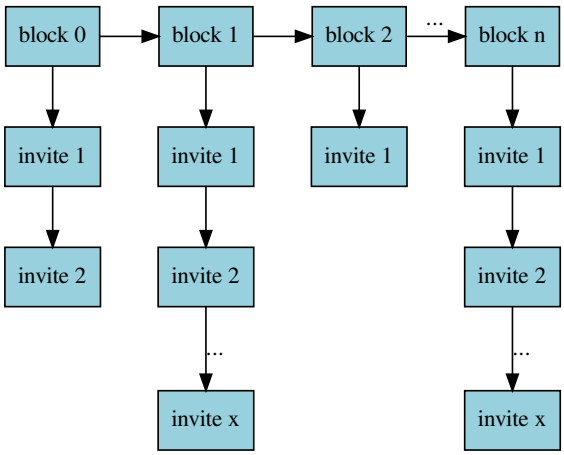

Figure 4: Decentralized Invite Generation

When the protocol decides to generate and distribute an invite, it first selects one of the lotteries by using the previous block hash as a seed. Each lottery has a certain probability of being selected, with the $SubCGS$ lottery being selected 50% of the time, the one-time award lottery 40%, and the remaining 10% is distributed via the random lottery.

$$seed = sha256(previousBlockHash)$$

$$hash = seed$$

$$lottery_0 = siphash(seed) mod 3$$

$$hash = sha256(hash)$$

$$lottery_1 = siphash(hash) mod 3$$

$$...$$

One the lottery is selected, it executes it to get an address to reward with an invite.

4.6.2 The SubCGS Lottery

This lottery is executed just as the $SubCGS$ lottery is for the ambassador rewards. The higher your $SubCGS$ the more likely you will be given an invite. The reasoning behind this is that those successfully growing the community should recieve more invites.

4.6.3 The One-Time Reward Lottery

This lottery is fairly straight forward. It looks at all candiates which have not previously one an invite and gives them one. Since you have to burn an invite to be able to get one in this lottery, making new addresses for the sake of participating does not make much sense and cancels out any benefit. The lottery is performed as a simple search.

Listing 2: One-Time Reward Algorithm

```
for candidate in  candidates
    next if candidate has already won once.
    next if candidate has more than 2 invites already.
    mark candidate
    return candidate
```

The reasoning behind this lottery is that new users should get an invite because they need a kick start to grow their own network.

4.6.4 The Random Lottery

It takes all candidate addresses in creation order and samples them using the previous block hash as a seed and successive hashing to determine the winners.

The first address is determined by taking the previous block hash and then running it through siphash [10] to determine a 64-bit random number.

$$seed = sha256(previousBlockHash)$$

$$rand = siphash(seed)$$

That amount is modded with the total confirmed addresses to pick the first address.

$$firstAddress = addresses_{(rand \mod total)}$$

Success winners are selected by repeatedly hashing combining the address with the previous hash to get a new value. The initial hash being the previous block hash.

$$itemHash = sha256(previousItemHash, address)$$

$$rand = siphash(itemHash)$$

$$address = addresses_{(rand \mod total)}$$

An attacker can try to generate a block that provides them the desired random number. Mathematically, this would take dramatically longer than it would take the community to mine a valid block. For an attacker to have a modicum of success, they would have to have more computational power than the rest of the network. Which, in of itself, creates too much centralization for any blockchain to be safe. This vector of attack would have to be repeated every time a valid block was mined (approximately every minute), which further decreases its likelihood of success.

Listing 3: Decentralized Invite Algorithm

```
invites = empty set
number of invites = average used in last X blocks per minute
hash = previous block hash
for i in number of invites
    rand = siphash64(hash)
    index = rand % total confirmed addresses
    address = address[index]
    invites.add(address)
    hash = sha256(hash + address)
```

4.6.5 Distribution Rate

The number of invites generated is based on the usage of invites within the last day. The invite distribution algorithm looks at the number of blocks that would have comprised the previous day (assuming 1 block issued per minute). The algorithm self-regulates in this way so that there are never too many or too few invites in the system at any given time. The system has an inherent minimum of two invites per ten minutes.

We can illustrate this through an example. If, in the last day, users have sent 7 invitations per block on average, that would mean that the current block should reward community members with 7 new invites.

4.7 Community Beacons

The beacon and invite mechanisms provide a base platform to build exclusive communities and authentication systems. People can create a community beacon which has an address just like a normal beacon. The community beacon is an extension of the existing beacon and contains a badge script.

Property	Description
Badge Script	Script which assigns a badge
Inviter Address	The address you were invited by.
Address	The address you want to beacon.
Public Key	The public key which to sign the beacon.
Signature	The signature produced by the public key.
Alias	An optional 20-byte field that can be used to assign an alias to an address.

Activating a community beacon is just like another beacon and requires an invite. However, sending Merit to the address beaconed requires the additional badge script to run successfully for the transaction to be considered valid. This extra validation provides opportunity to restrict beacons in various ways.

4.7.1 Badges

Badges are assigned to addresses when an address sends Merit to the community address and the badge script validates the transaction. The creators of a community beacon can query the Merit system whether the badge has been assigned or not to an address.

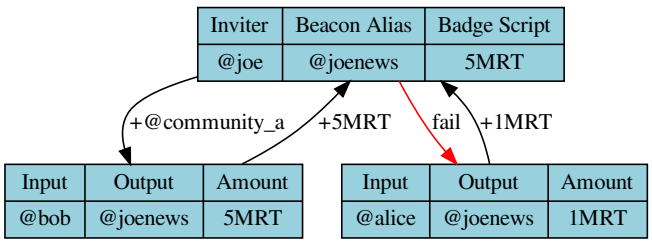

Figure 5: Community Beacons and Badge Assignment

Badges allow people to create exclusive communities. For example, a website which wishes to charge for content can allow users to send a specified amount of Merit to the website's alias, and the sender will automatically be assigned the community badge. The website operator can then challenge the user to prove they are the owner of the address by making them sign some data with their private key. Once the address owner verifies their identity, the operator can check whether the address has the correct badge and allow them to access the website. Another example is a private venue wanting to sell tickets. They can create a community beacon and ask patrons to send Merit to the beacon to get a digital ticket in the form of a badge. Software and hardware creators can provide authentication systems which build on the alias and badge system to replace traditional tickets, subscriptions, and other services that have payment dependant features.

4.7.2 Badge Scripts

Merit builds upon the Bitcoin scripting system. Transactions and badge scripts use the same virtual machine which is stack based and is inspired by the Forth programming language [12]. However, badges require new instructions for the virtual machine (called opcodes) to be useful for building communities and the desired use cases outlined above. The Bitcoin scripting system is designed so that scripts have minimal context and are not transaction aware. For Badges to function, we need to create new opcodes which can query the blockchain or inspect different parts of the transaction like the inputs and outputs.

Opcode Name	Inputs	Output	Description
INPUTADDRESS	Index	Address	Returns the address at input index
INPUTCOUNT	NA	Count	Returns the number of inputs in the transaction
HASBADGE	Address,Badge	Bool	Returns true if the address has the badge specified

With Merit transactions, the scripts work by combining an output script with the coin's public script for validating an output. Badges work the same way but add the badge script with an empty stack as additional validation. Vaulting, described below, introduces even more opcodes which are useful for badges as well.

An obvious downside to adding powerful opcodes that can query the transaction or blockchain is that validation time must necessarily increase. Great care must be taken to provide an upper bound in time that a transaction can take to validate. This upper bound is not defined here and will remain implementation specific.

4.7.3 Badge Transaction Validation

The beacon's badge script validates whether a transaction to the address of that beacon is valid. This validation happens to each transaction to the beaconed address. It executes the script for each output in the transaction that matches the address and uses the outputs script as input to the badge script. The badge script can ignore the stack or use it in constructive ways.

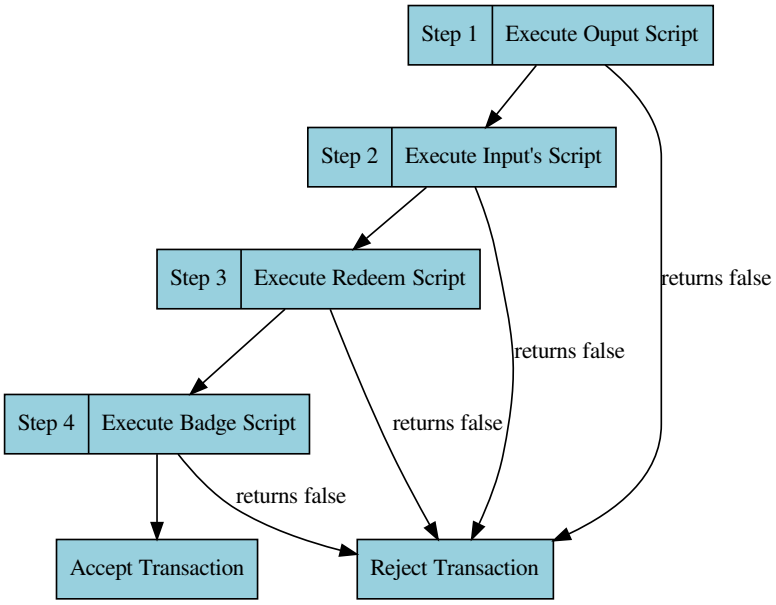

Figure 6: Badge Validation State Machine

4.7.4 The INPUTADDRESS Opcode

Opcode Name	Inputs	Output	Description
INPUTADDRESS	Index	Address	Returns the address at input index

The INPUTADDRESS opcode extracts the address from the input at the index specified and pushes it to the top of the data stack. The output is bytes of the RIPEMD-160 hash which represents the address.

4.7.5 The INPUTCOUNT Opcode

Opcode Name	Inputs	Output	Description
INPUTCOUNT	NA	Count	Returns the number of inputs in the transaction

The INPUTCOUNT opcode pushes the total number of inputs in a transaction to the top of the execution stack. INPUTCOUNT allows the script writer to validate specific properties about the transaction and a way to compute what the index is of the last input. This opcode is useful in concert with the INPUTADDRESS opcode. For example, the user may want to say that the last

input must have a particular badge. They can figure out the last input index using the INPUTCOUNT and retrieve the address using INPUTADDRESS. They can then use HASBADGE to validate that the input address contains the badge requested.

4.7.6 The HASBADGE Opcode

Opcode Name	Inputs	Output	Description
HASBADGE	Address, Badge	Bool	Returns true if the address has the badge specified

The HASBADGE opcode expects there to be two elements on the execution stack. The top of the stack is an address, and the second is the badge. It pops off the inputs and puts True on the top of the stack if the address has the badge specified. This opcode is particularly challenging to implement because it must do a query about an address on the blockchain. An appropriate implementation will use an index where the search is no longer than $O(logN)$ in complexity. Since $logN$ is small for any reasonable size of N (anything with 64bits), this should provide adequate performance. Also because that badge scripts are run last after all the other transaction validation, most DDOS [13] attacks should be mitigated. In the event of too many failed transactions, the node software can ban further transactions against a beacon by the input addresses. Since Merit is invite only, addresses are a scarce commodity, and the cost of a ban is not insignificant. The HASBADGE opcode is particularly useful to allow more complex community relations and also tiered pricing to those wanting compensation for joining a community. A user may create a @newstier1 and a @newstier2 community where joining the second requires being part of the first.

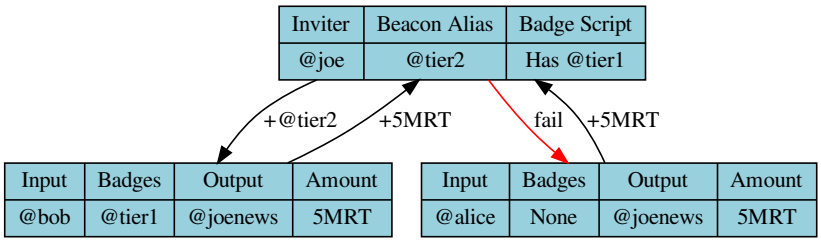

Figure 7: Badge Tiers

5 Safety

Safety is a property that users have discovered they value greatly. Many third-party services now provide vaulting a feature on top of existing currencies. Merit provides a decentralized approach to vaulting by implementing a simplified form of covenants. Merit extends the existing Bitcoin scripting system with new opcodes which enable important safety features such as vaulting that doesn't require a trusted third-party. In addition to new scripting opcodes, Merit's unique beaconing system.

5.1 Beacons

Merit does not allow non zero amount transactions to addresses that are not beaconed. This safety feature protects users against common mistakes of mistyping or copying addresses. Users can't mistakenly send Merit to an invalid address preventing a source of currency leakage and misuse. It brings a vital safety feature from the bank world in a decentralized way. All forms of addresses must be beaconed including scripts.

Beacons also allow the correct attribution and enable patronage and a way for users to support people and organizations they care about since users can choose the inviter address of a beacon.

5.2 Vaulting

Merit builds upon the Bitcoin scripting system by introducing several new opcodes which allow building decentralized vaults. The existing Bitcoin scripting system is designed so that scripts have minimal context around their execution. Merit chose to extend Bitcoin's scripting system because it is well understood by the broader community, well tested, and also not Turing complete. The latter property significantly improves the safety of the overall system. Other cryptocurrencies that have Turing complete virtual machines like Ethereum have been shown to be too error-prone and easily exploitable. By building on Bitcoin's Forth-like stack-based virtual machine, the overall system is more natural to reason about and control.

5.2.1 Transaction Aware Scripts

Bitcoin scripts have no insight into the outputs of a transaction. Merit improves upon Bitcoin's scripting system by providing OPCODES which give script context about the transaction the script is executing in. We introduced three new OPCODES which allow vaults to work and some additional OPCODES to reduce the instruction count for vault scripts minimizing their cost.

Opcode Name	Inputs	Output	Description
CHECKOUTPUTSIG	NumAddresses Address1 ... AddressN OutputIndex NumParams Param1 ... ..ParamN	Bool	Validates an output at index OutputIndex
OUTPUTAMOUNT	OutputIndex	Amount	Returns amount at output index OutputIndex
OUTPUTCOUNT		Num	Returns number of outputs in the transaction
NDUP	Num		Duplicates the next N elements on the stack
NDROP	Num		Removes the next N elements from the stack
NTOALTSTACK	Num		Pushes the next N elements to the alt stack and from the stack
NFROMALTSTACK	Num		Pushes the next N elements to the stack from the alt stack

These new opcodes can be used to construct a vault script that only allows funds to be transferred to a whitelist of addresses at a specific rate limit.

5.2.2 Simple Vault Script

The following vault script provides an illustrative example.

Listing 4: Simple Vault Script

```
//Stack
<signature>
<public key>
<spend limit>
<address 1>
<address 2>
<number of addresses>

//Script
NTOALTSTACK
TOALTSTACK
DUP
TOALTSTACK
CHECKSIGVERIFY      //check key used to secure the vault.
FROMALTSTACK
FROMALTSTACK
DUP
0
OUTPUTAMOUNT
GREATERTHANOREQUAL //make sure not too much is spent.
VERIFY
0
0
NFROMALTSTACK
NDUP
NTOALTSTACK
CHECKOUTPUTSIGVERIFY //validate that amount is sent
                     //to only addresses of the whitelist
NFROMALTSTACK
DEPTH
1
's'
1
CHECKOUTPUTSIGVERIFY //make sure change is put back in the vault
                     //the 's' above means the hash of this script.
                     //therefore, the vault must be the same.
2
OUTPUTCOUNT          //make sure there are no other outputs
                     //in the transaction.
EQUAL
```

5.2.3 The CHECKOUTPUTSIG Opcode

Opcode Name	Inputs	Output	Description
CHECKOUTPUTSIG	NumAddresses Address1 ... AddressN OutputIndex NumParams Param1 ... ..ParamN	Bool	Validates an output at index OutputIndex

The CHECKOUTPUTSIG opcode allows a script to validate that a destination satisfies specific requirements.

It first validates that the addresses specified in the output match the ones listed in the opcode. This output check allows implementing whitelists where funds can only be transferred to a restricted list of addresses. CHECKOUTPUTSIG allows users to create individualized coins which limit the value of stealing their private keys.

This output check combined with *Parameterized Script Addresses* allow a script to make sure an output is the same *kind* of coin, which is crucial for implementing features like vaults. It also allows coloring coins which can construct sub currencies that have unique properties.

If the destination is a *Parameterized Script Address*, you can create a parameter-mask which checks that the output matches against the mask. Each parameter can match exactly or be changed.

SELF is a special address which makes sure that the destination address matches the script's address. In other words, you can limit a coin to transfer only to itself. This isn't particularly useful by itself because that would lock the funds into a coin that can never be spent. However, this feature combined with *Parameterized Script Addresses* provide a powerful way to contain a coin and control how it can be transferred.

5.2.4 Parameterized Script Addresses

To support READONLY parameters to scripts, Merit created a new address type called Parameterized Pay-to-Script-Hash which provides a list of read-only stack items that are part of the unspent output. These are then appended to the scriptSig after the redeem script is expanded. It is essential to lift some parameters out of the script directly because the $CHECKOUTPUTSIG$ opcode allows outputs to change specific parameters. Merit provides more complex vaults which have two or more keys. You can provide a spendable key to someone with a rate limit while keeping a master key that allows you to change certain aspects of the vault. For example, if the spend key is compromised, you can change the vault with the master key preventing further theft.

These safety features are provided in a decentralized way and vaults can function without third-parties.

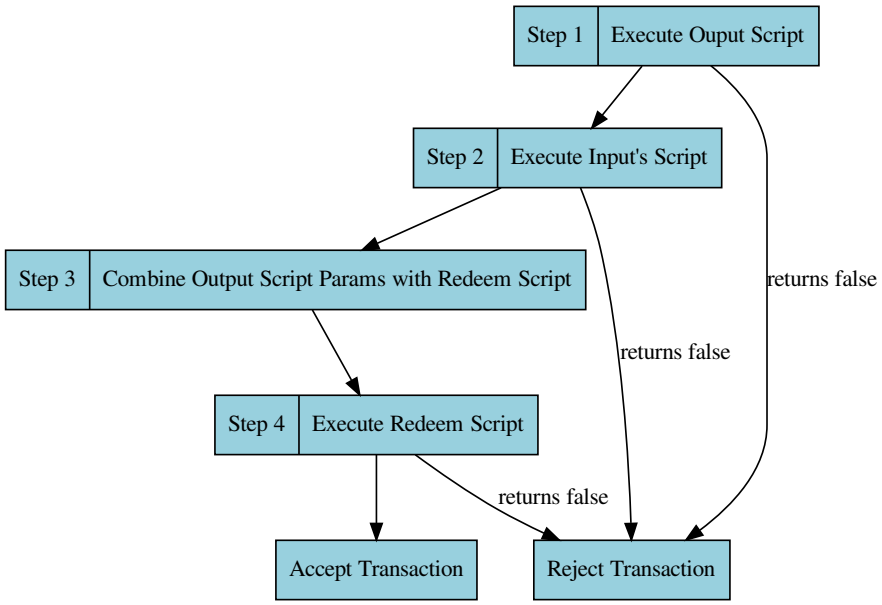

Figure 8: Parameterized Script Validation State Machine

5.3 The OUTPUTAMOUNT Opcode

Opcode Name	Inputs	Output	Description
OUTPUTAMOUNT	OutputIndex	Amount	Returns amount at output index OutputIndex

Another useful property of vaults is the ability to rate limit the amount of Merit that can come out of. Merit has added an opcode called OUTPUTAMOUNT which returns the amount of Merit being sent to an output at a specific output index in the transaction.

The opcode takes the output index from the top of the stack, consumes it and adds the amount of Micros (The smallest division of Merit) at that specific index. If the index is out of bounds, an error is returned, and the transaction is not valid.

5.4 The OUTPUTCOUNT Opcode

Opcode Name	Inputs	Output	Description
OUTPUTCOUNT		Num	Returns number of outputs in the transaction

The OUTPUTCOUNT opcode returns the total number of outputs in the transaction and does not have any input.

This opcode is critical in being able to implement vaults and other safety features because it can be used to limit the number of outputs a transaction can have. Without this opcode, rate limiting and other vault features could not be possible. Rate limits would not be possible because you can always send Merit from a coin to an output that isn't checked by the OUTPUTAMOUNT opcode. By limiting the number of outputs, you can write a script to make sure all those outputs follow the rate limit. Similarly, features like whitelists would not be possible because someone could always transfer Merit to an output not checked by CHECKOUTPUTSIG. This simple opcode is, therefore, necessary to provide the safety features of Merit.

5.5 The NDUP Opcode

Opcode Name	Inputs	Output	Description
NDUP	Num		Duplicates the next N elements on the stack

The NDUP opcode duplicates the N entries on the stack. It allows the ability to have an array like semantic on top of the stack. Because vaults require the use of *Parameterized Script Addresses* which have arrays of parameters, this opcode allows vaults to be more straightforward and shorter, requiring fewer bytes. This lowers the code of vault transactions and also their storage on the blockchain.

5.6 The NDROP Opcode

Opcode Name	Inputs	Output	Description
NDROP	Num		Removes the next N elements from the stack

This NDROP opcode is the opposite of the NDUP opcode and removes N elements from the stack. The motivation behind this opcode is the same as NDUP, to allow array semantics and also simplify and shorten vaults and similar scripts that deal with *Parameterized Script Addresses*.

5.7 The NTOALTSTACK Opcode

Opcode Name	Inputs	Output	Description
NTOALTSTACK	Num		Pushes the next N elements to the alt stack and from the stack

Similar to Bitcoin, Merit scripts have two stacks, the main stack, and the alternate stack. The NTOALTSTACK opcode allows a user to transfer N items to the alternate stack in one command. It expects the number of items to be at the top of the main stack and will pop that many items off the main stack and push them to the alternative stack. And finally will push the count of items to the top of the alternative stack. This allows array-like semantics when working with the alternate stack.

The items pushed to the alternative stack will be in reverse order after this opcode is executed.

5.8 The NFROMALTSTACK Opcode

Opcode Name	Inputs	Output	Description
NFROMALTSTACK	Num		Pushes the next N elements to the stack from the alt stack

This opcode is the opposite of the NTOALTSTACK opcode. It excepts the top of the alt stack to be the count of items and then will pop them off the alternate stack and push them to the main stack. Executing NTOALTSTACK and then NFROMALTSTACK will return the main stack to its state before NTOALTSTACK was called. This allows array-like semantics when working with the alternative stack.

6 Usability

One reason users are pushed towards trusting and using third-party services is that the usability of many cryptocurrencies is weak and is a step back from many of the conveniences of state-backed currencies. Merit is built with the idea that the hard problem of usability must be tacked. Improving the usability of decentralized systems is far harder than centralized systems and requires significant time and investment.

Some usability enhancements have already been discussed, such as beaconing which allows address verification and prevents users from accidentally sending funds into the ether. We talked about vaults which provide an more natural way to secure and manage funds and make common key management mistakes less tragic.

One ease of use feature that was introduced by Paypal is the ability to send funds to people who are not yet on the platform via private communication channels. Merit provides a decentralized system to send funds to new users of Merit.

6.1 Frictionless Transactions

Merit provides a feature to quickly and easily send Merit to those that are currently not using Merit. It both sends merit and invites that person in one step. Merit has a new opcode called EASYSEND which allows a user to put

funds in a decentralized escrow. The funds can be canceled and sent back to the sender, and are unspendable by the receiver after a certain amount of time. The sender sends the recipient a secret key and their public key. The recipient then can use that information to get the funds.

Opcode Name	Inputs	Output	Description
EASYSEND	NumKeys Key1 ... Keyn MaxDepth Signature	Bool	Allows accepting funds by any keys listed. First key is can recieve funds after the timeout.

The motivation for a new opcode is to limit the fee of sending these kinds of transactions by making them smaller. This building block can be used by a light wallet to construct URLs which automatically accept funds once the light wallet app is loaded. A light wallet is software which allows a user to work with the system without having to have the entire blockchain on their computer.

6.1.1 Merit URIs

Transaction URIs can be sent via any private channel such as SMS or email to people who are not currently using Merit.

```
merit:?sk=ABC123&pk=DEF456&t=1000
```

Param	Description
sk	Secret Key
pk	Public Key of Sender
t	Timeout in block height

The parameters are designed to take up the least amount of characters while still being recognizable.The funds are available to them for a certain amount of time where they can download or install the wallet software and accept the funds. The sender can always cancel the transaction and get the funds back before or after the time limit has expired.

For cases where operating systems will not support the Merit URI schema, an alternative proposal is to use standard URLs.

```
https://<host>/recieve/?sk=ABC123&pk=DEF456&t=1000
```

However, great care must be taken by the host not to log parameters to these requests. This URL approach might be implemented by light wallets or web wallets which want to drive new users to use their implementations.

7 Scalability

The most popular cryptographic currencies are limited in their scalability in several ways. The massive centralization of mining has introduced many single points of failure into the decentralized system. Also, the transaction rates of Bitcoin and Ethereum are very low and have led to high transaction fees during times of congestion [16]. These high fees have made these systems limited in their daily utility.

Merit tackles the centralized mining problem by embracing an ASIC resistant mining algorithm based on the Cuckoo Cycle [2]. Merit improves the transaction rate over Bitcoin in two stages. First by providing a frequent block-time with large blocks. And then later, using a generalization of the blockchain called the block fabric.

7.1 The Cuckoo Cycle: Merit's Proof-of-Work

Merit uses a memory hard asymmetric proof-of-work called the Cuckoo Cycle [2]. This algorithm constructs a bipartite graph from a hash seed where the proof-of-work is a 42 length cycle between the nodes in the graph. Verifying this cycle is simple while finding it is hard. The performance of finding a cycle is correlated to the memory bandwidth of the underlying hardware. Since memory technology is competitive and a commodity, the Cuckoo Cycle should allow people with conventional hardware to compete. Building an ASIC to increase compute power does not improve the performance of the algorithm since it is memory bandwidth bound.

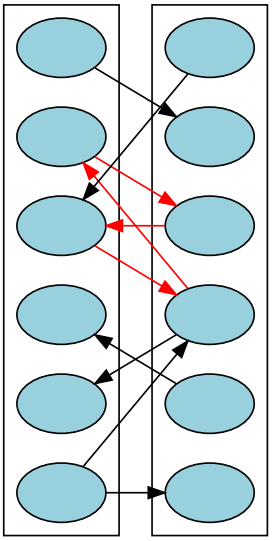

Figure 9: Bipartite Graph with Cycle of Length Four

Using the Cuckoo Cycle enables an extra degree of freedom in adjusting the difficulty. In addition to the bitcoin $nBits$ difficulty parameter which specifies how many zero bits the hash must have to be considered a valid proof-of-work, Merit also introduces $edgeBits$ which specifies the minimum size of the graph that needs to be computed. Currently, at the time of writing this, a single proof attempt where $edgeBits = 26$ takes about 700MB of ram. This is reasonable enough that the mining software can run on a mobile device.

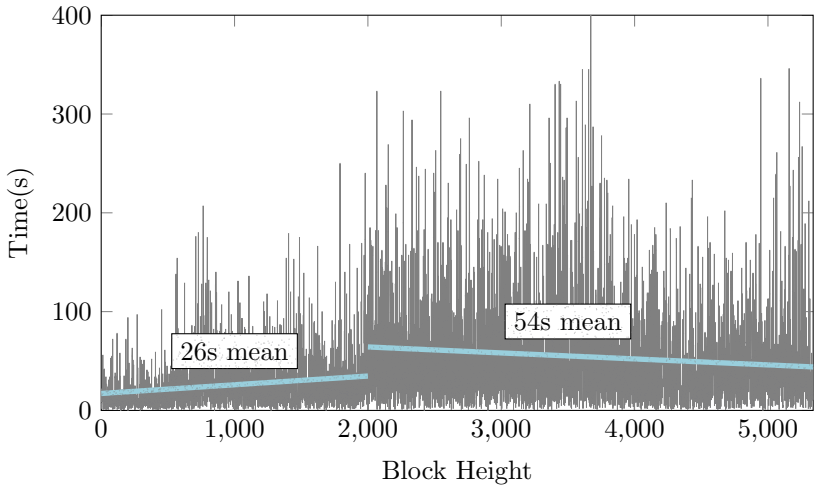

Figure 10: Empirical Merit Block Time

The graph above depicts time between blocks. The adjustment of the difficulty can be seen given the two regression lines, where the adjustment happened around block 2000. The difficulty adjustment attempts to maintain a time between blocks of one minute on average. However, the times blocks are solved is random and the actual times fall within a very specific distribution.

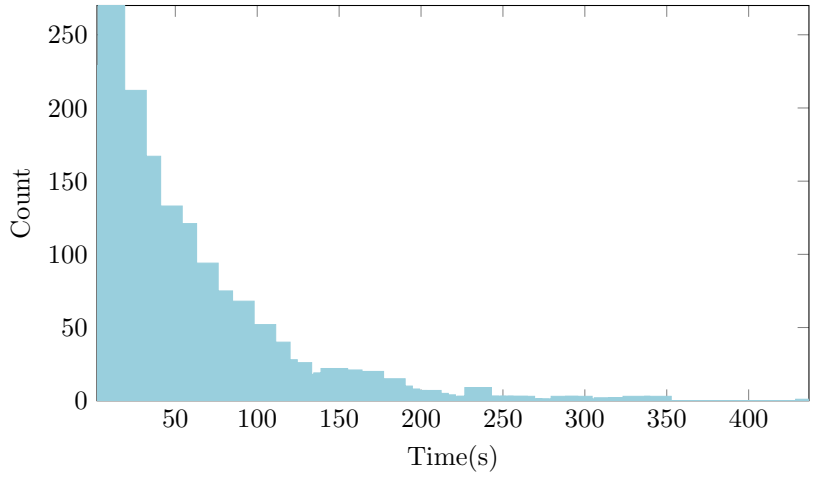

Figure 11: Empirical Merit Blocktime Histogram

The time between blocks follows a Poisson Point Process [3], and the mining

algorithm can be modeled as an exponential distribution. The standard deviation computed from the actual mining process is close to the mean in both before and after adjustment.

$$std = \sqrt{\frac{1}{\lambda^2}}$$

Before the adjustment, the mean time was 26 seconds with a standard deviation around 28. After the adjustment, the mean time went to 54 seconds with a standard deviation around 53.5. The standard deviation from the empirical data matches what you would expect from an exponential distribution, which is close to the mean.

This is the same distribution as mining Bitcoin [4] and should provide the same lottery dynamics. But because the block frequency is a minute instead of ten minutes, and the block size is 16MB instead of 1MB, then a person with a commodity machine has a fighting chance to win a block since the probability of finding a valid Cuckoo Cycle within a short amount of iterations is higher. There is no ASIC advantage where a single machine may have many orders of magnitude performance increase over another piece of hardware.

7.2 Bigger Frequent Blocks

In the first stage, Merit tackles the transaction rate problem by offering block times of 1 minute with a 16MB block size, providing a potential 160x improvement in the transaction rate over Bitcoin.

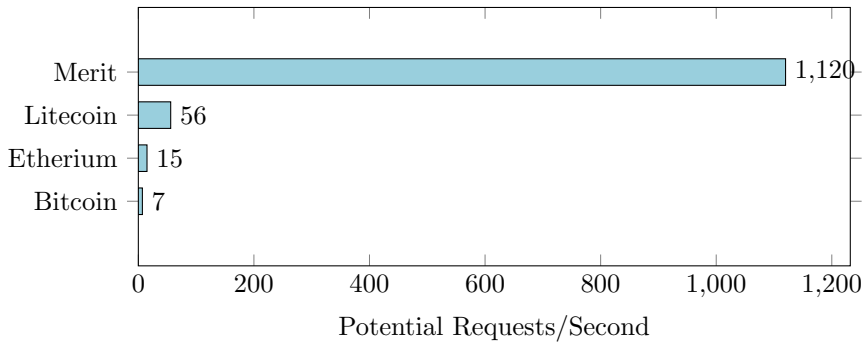

Figure 12: Stage One Transaction Rate

7.2.1 Bandwidth

In practice, the maximum block size might not be reached because of bandwidth limitations in the network. Research done in the Bitcoin Unlimited paper [17] looks at the impact of various block sizes. Miners are at risk of creating orphaned blocks when as the size of blocks increases block because it takes some time for

a block to propagate across the network. The larger the block size, the higher the chance the block will compete with other blocks mined during the time it takes to propagate.

Therefore increasing block size beyond the network's capabilities to propagate blocks provides no improvements over the transaction rate. At some point, miners won't put more transactions because the risk of creating stagnant blocks becomes too large. A different method of increasing performance must, therefore, be considered.

7.2.2 Lightning Network

The Bitcoin lightning network attempts to offload transactions from the blockchain to a network of channels. The idea is that channels only need to announce final results of all transactions on the blockchain once they are closed. Any two parties can send transactions to each other by routing the transactions through a series of channels. Unfortunately, there are several issues with the lighting network which provide real-world limitations.

- Opening and closing a channel requires staking coins in a funding transaction.
- Routing a transaction requires intermediaries to have the liquidity necessary to route transactions.
- The system is susceptible to DDOS attacks.
- The system is susceptible to fraud and may require users to use watchtower nodes.
- The complexity of creating and maintaining a channel may mean it is appropriate for large institutions.

Merit is still evaluating whether it will incorporate lightning network features into its system.

7.3 Block Fabrics: A Generalization of the Nakamoto Blockchain

A more dramatic approach needs to be taken to scale Merit for stage two because increasing block size and frequency won't scale to support a sizeable worldwide user base. The larger blocks and shorter block time will improve transaction rates over Bitcoin and Ethereum, but is a fundamentally limited approach because the speed of the network is only as fast as the speed of one machine on the network. To achieve real scalability, we need to design a system that has multiple writers by sharding the address space and converting a one-dimensional database into a two dimensional one.

The crucial insight is that the Nakamoto Blockchain is a degenerative case of what we call block fabrics, which are inspired by thousands of years of tradition from the craft of weaving which create a form and structure of time.

7.3.1 The Weave and Warp

Merit implements an algorithm inspired by the weaving of fabrics. Block fabrics have many threads aligned in one direction in parallel called the *Warp* and these threads have a single thread woven through it back and forth called the Weft. In other words, many parallel strands are held together by a single thread weaved through them over time. Instead of calling this single thread the Weft, we will use the term the Weave to highlight the relationship with time better.

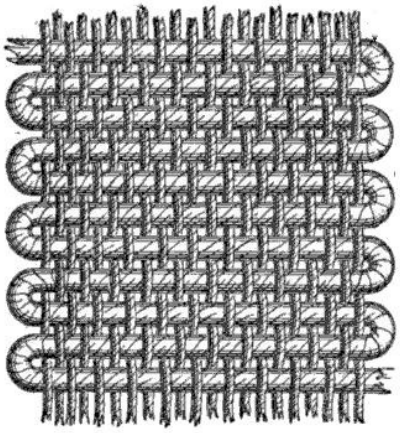

Figure 13: Weaving Fabric [19]

7.3.2 Warp Chains

Merit's sharding algorithm divides the address space into shards called *Warps*. Each *Warp* is a blockchain and has a single writer for a period of time called the leader. If we think of the Warp's existence in the platonic world, the Warp has existed from the beginning of time and will exist to the end of time. In some sense, the Warp is independent of time similar to how a real Warp on a loom is fixed before the weaving process begins, Yet the fabric is wholly defined by time when it is woven.

Each miner competes to become the leader of a Warp by mining blocks on the *Weave*. An explanation of how the *Weave* functions is described later. Each *Warp* has the following properties.

Property	Description
Blockchain	Each Warp is a blockchain.
Single Writer	A Warp has a single leader at any given time.
Prefix Range	Identity of a Warp is its address range.
Many Warps	Number of Warps at any given time is defined.
Have Transactions	Blocks in a Warp have transactions, invites, and beacons.
No PoW	Blocks do not have a proof-of-work.
Signed	Blocks are signed by the leader.

Each warp block Has the following properties defined in the block header.

Property	Description
Previous Block	Hash to previous block in the warp.
Weave Block	Hash to weave block.
Merkel Root	Merkel Root hash describing the set of transactions
Signature	The signature of the leader of the warp chain.
Public Key	The public key of the leader of the warp chain.

Notice there is no nonce or proof of work in this chain. The leader signs each block and is allowed create as many blocks as possible that pass the address range validation defined in the referenced weave block.

7.3.3 Prefix Range

A Warp has a continuous address range with N bytes in size.

Property	Description
Prefix Size	Number of bytes to match.
Start	Start of byte range.
End	End of byte range.

The bytes that match against the address range are any bytes that are a permutation between the start and end of the range. Each Warp has a mutually exclusive address range assigned to it via the *Weave* process described later.

7.3.4 Warp Assignment Overview

Each *Warp* is a blockchain where each block contains transactions, invites, and beacons that are assigned to it. The *Prefix Range* of the Warp is used in the assignment algorithms to match the asset to the Warp. The assignment logic for beacons, invites, and transactions is outlined in the table.

Type	Assignment Logic
Beacon	If the Beacon's address matches the Warp prefix
Transaction	If one of the input's addresses matches the Warp prefix
Invite	If one of the input's addresses matches the Warp prefix

7.3.5 Invite and Transaction Assignment

Invites and Transactions have the same structure, and therefore their assignment to a Warp follows the same logic. Merit inherits the transaction structure of Bitcoin. Each Transaction has one or more inputs and one or more outputs. Each input references a previous output by transaction id and index.

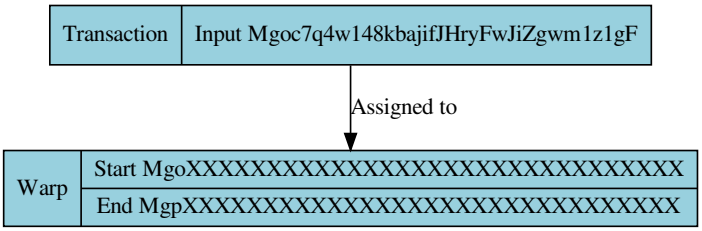

Figure 14: Beacon Assignment

Each transaction input must have an address associated with it given the rules of the Merit invite system. A transaction is assigned to a Warp if it's input's address matches a Warp's prefix. Because most transactions have multiple inputs, the transaction is split into many transactions.

7.3.6 Beacon Assignment

It is vital that the Beacon ends up in the same Warp as the Invite that is used to activate the Beacon. Each Beacon has an address which it is advertising, and it is this address that an invite is sent. The Beacon is matched to a Warp by matching the Beacon's address to the Warp's address range. Since Warps are guaranteed to have mutually exclusive prefixes, then the Beacon will end up in the same Warp as the Invite sent to it.

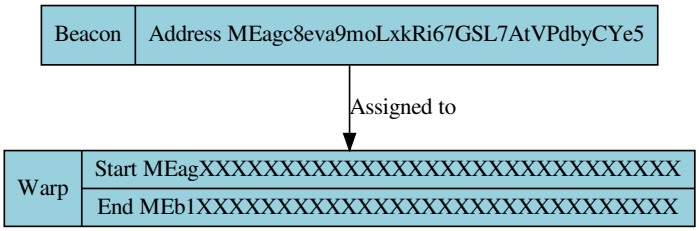

Figure 15: Beacon Assignment

7.3.7 Transaction Splitting

Transactions with more than one input are split into many transactions with the same outputs. If a transaction has N inputs, it will be split into N transactions. The original transaction fee is split among the Warps and rewarded to the Warp leaders given the Warp fee rules. The original transaction is considered complete when all splits appear in blocks on their assigned Warps.

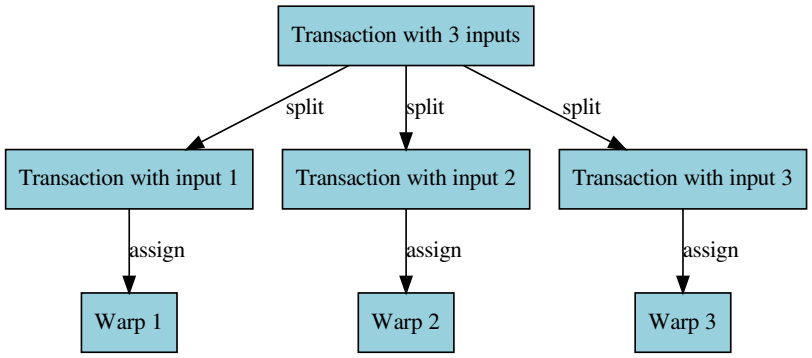

Figure 16: Transaction Assignment

7.3.8 The Weave Chain

Miners compete on the Weave chain using the Cuckoo Cycle proof-of-work. The Weave can be thought of as the main blockchain. It weaves through each Warp in a deterministic order by partitioning the whole address space into N warps. The size of N must be necessarily a power of two. This means there can be, 2,4,8,... Warps. Each block in the Weave chain has three parts...

1. Warp Owner Announcement.
2. Merit Coinbase Transactions and Ambassador Rewards.
3. Invite lottery rewards.

The weave block has the following data in the header.

Property	Description
Previous Block Hash	Hash to previous block in the warp.
Merkel Root Hash	Merkel Root hash describing the set of transactions.
NBits	The number of leading zero bits the proof-of-work must have.
EdgeBits	The number of bits used to describe an edge cycle.
Nonce	An integer that is incremented to change the block hash.
Cycle	The proof-of-work in the form a Cuckoo Cycle.
Warp Number	Warp number.
Total Warps	The total number of warps.
Signature	The signature of the leader of the warp specified.
Public Key	The public key of the leader of the warp specified.

The weave blocks require a proof-of-work and are signed by the winning writer of the warp block. Which warp they can write to is specified by the warp number that is deterministically selected in a round robin fashion.

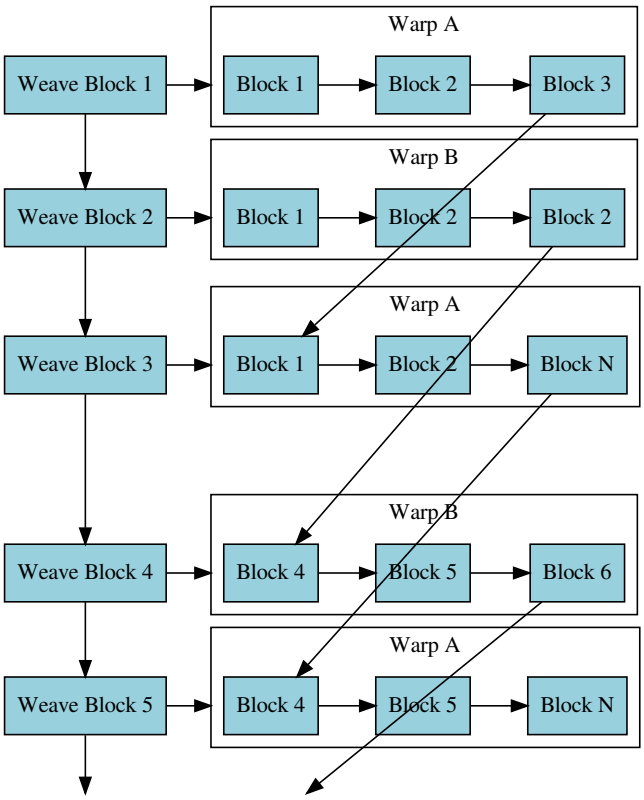

Figure 17: Weave and Warp

7.3.9 Warp Owner Announcement

When miners successfully mine a block on the Weave chain, they take over being the write to a Warp that is deterministically determined by taking the next Warp in the set of N Warps that are assigned at the time. If we assign each Warp an index W into the set of N, and any given block B_i returns the assigned warp index W with the function $A(i)$. The Warp assigned to block B_i is

$$A(i) = A(i-1) + 1 \mod N$$

.

7.3.10 Address Space Partitioning

Just like Bitcoin, Merit addresses are a RIPEMD-160 hash function which is a 160-bit hash. This is a huge space and is broken up into any practical number of pieces of equal shards. The size of each shard can be easily specified by the number of significant bits required to describe the start and end of each shard. Since we don't expect the number of shards to be too significant at any given time, then the address range can be compactly stored using the minimum number of bits required to define the start and end of the range.

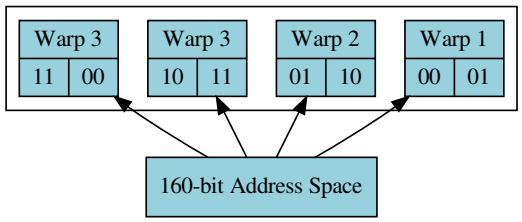

Figure 18: Four Warp Partitioning

7.3.11 Dynamic Warp Adjustment

The number of Warps can be adjusted dynamically by looking at the transaction rate over the last X blocks. We can decide what *At Capacity* is per Warp, and use that to maintain enough Warps never to surpass this capacity. These attributes can be fixed to create a deterministic algorithm for determining the number of Warps at any given block on the Weave chain. When the number of Warps N increases from one block to the next, the new Warp assignment will assign only part of the previous Warp. In this situation, the new Warp partially occludes a previous Warp. Each warp leader must watch the Weave blockchain to understand when their Warp is partially occluded. Any Warp blocks after the block referenced in the occluding Weave block must not have transactions that are occluded. This may require the occluded warp leader to abandon any blocks after the referenced block and start writing again.

7.3.12 Coinbase Transactions and Ambassador Rewards

Coinbase transactions, Ambassador Rewards, and invite generation occur on the Weave chain. This makes sense because these transactions have no inputs

and therefore cannot be assigned to a Warp. This rule also requires changes in the existing blockchain algorithm defined in stage one scaling where Merit is a single blockchain with 16MB blocks and one minute block time.

7.3.13 Warp Fee Payout

To incentivize future leaders of a Warp to pick the latest block written by the previous Warp leader, the previous leader will give 60% of the fees they accumulated to the next leader and keep 40%. This 40/60 split is inspired by the Bitcoin-NG [20] paper. The Bitcoin-NG paper assumes an attacker is bounded by 1/4th of the mining power. However, because Warps partition the address space, and therefore the fee distribution, the attacker has a smaller incentive to misbehave because the fees from all transactions are distributed among all leaders. The attacker must have a sustained attack over the period of $N * 2$ Weave blocks where N is the number of Warps to capture 100% of the transaction fees for an N block period.

7.3.14 Fabric Analysis

Looking back at the two concepts, we can summarize by saying that the Weave assigns leadership of writing capability to a Warp to the miner who wins a Weave block. They can write to the Warp until another leader of the Warp takes over. The assignment of the Warps is well defined and ordered where the time between Warp leadership changing hands is regular. For example, if the number of Warps available is N, then a Warp leader will expect to lose leadership of the Warp in N minutes. This is because the time between blocks on the Weave chain is one minute.

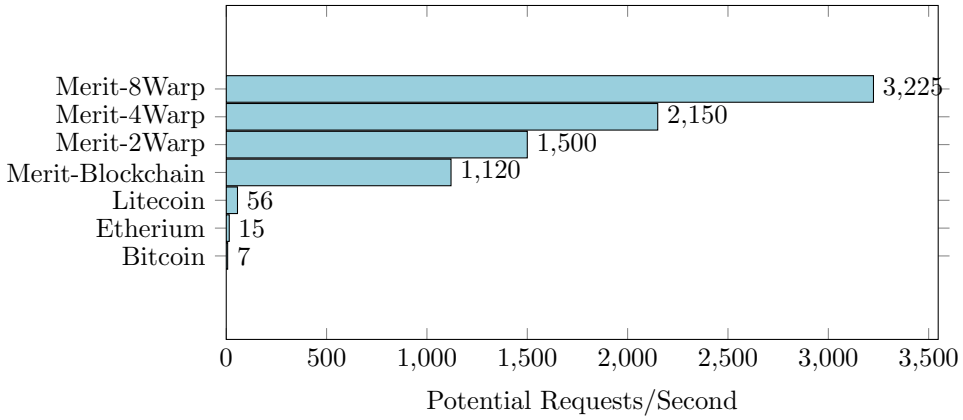

Figure 19: Stage Two Transaction Rate

There is a constant balance between increasing throughput or reducing la-

tency. Merit attempts to reduce latency by providing fast minute block times on the Weave blockchain. At the same time, throughput is increased by having multiple writers committing transactions on the blockchain. A downside is that transaction on a given Warp are confirmed for every N blocks where N is the count of Warps. This is because transactions from a particular reader are committed to the blockchain once the next leader for that Warp acknowledges the blocks of the current leader by mining a block no the Weave chain. This means that latency of confirming transactions is highly predictable and is a function of the number of Warps N and the time between blocks on the Weave. As an illustrative example, if N is two, then the latency when a Warps transactions are committed is two minutes. As the system reaches capacity, N would be doubled to four and therefore the confirmation time would increase to four minutes. When N reaches 64, we end up with a confirmation time around an hour. The relation between throughput is geometrically defined via the space-time relation of the Weave and Warp chains.

As an illustrate example, if we say each Warp leader is capable of writing 1K split transactions per second, and likely 100% of those transactions created from a transaction with at least two inputs, then we can say the overall transaction throughput is $1/2$ of the Warp transaction throughput. So with 2 Warps, we have an overall rate of 1000 transactions per second because $1000 = 2000/2$. Four Warps would double the transaction rate to 2K transactions per second. It is easy to see how only having a few Warps can increase the transaction rate significantly. However, it is important to note here that this assumes transactions with only two inputs. With more substantial transactions, the throughput will not double so cleanly. The dynamic warp adjustment algorithm should compensate for this and provision enough warps to handle the required transaction volume. An analysis of the average input size once Merit has grown would provide a chance for a better analysis.

The obvious take away here is that eventually for any machine to have a set of all the unspent outputs (UTXOs), it would eventually need to index all the transactions in the system.However, this process can be much faster than the original block generation process of the Warp leaders. This is because the Warp leaders do more work because they have to decide which transactions to put in a Warp.

8 Summary

Merit has aggressively focused on community, ease of use, safety, and scalability in an attempt to tackle the centralization and trust in third-parties that have corrupted the vision of the cryptographic currency community.

Merit recognizes that there is a wide variety of users of the system and introduces incentives to those ambassadors of the system that go beyond mining. By having an Ambassador Tree, we allow users to support people and organizations they love without having to understand and manage complex software systems.

Glossary

address An address refers to short representation of a public key. The public key is hashed with RIPEMD-160 and then base-58 encoded to create a succinct reference to the public key.. 44, 45

ambassador An ambassador is a user of Merit who invites others and helps build a strong community. They are represented in the system via a beacon.. 44

ambassador forest An ambassador forest is the collection of all beacons. Merit considers each beacon an ambassador because each person using merit can invite others.. 44

ASIC ASIC stands for application-specific integrated circuit. These are specialized machines which accomplish what was once a software task in hardware. The reason for building these is usually to improve the performance of executing that specialized task.. 1

beacon A beacon advertises an address and it's relationship with other addresses. It contains a reference to an inviter address, and beacons construct an ambassador forest. Beacons are signed and contain the corresponding public key.. 5, 44, 45

bearer currency A money whose ownership is controlled by the holder and there is no central record keeping of who owns what. In the cryptocurrency domain, this usually refers to the fact that proof of ownership is a signature with a private key, and if the bearer of the key loses said key, they lose the ability to prove ownership.. 3

Bitcoin A cryptocurrency created by Satoshi Nakamoto. Merit is based on Bitcoin.. 3, 45, 46

block A node in the blockchain. Each block references the previous block by the block's hash. The block is composed of a header and a ledger. The block hash is computed by hashing the data in the block header. The ledger contains transactions and beacons. 44–47

block fabric A generatlization of the blockchain that improves on scalability by dividing the problem space into weaves and warps.. 2, 30, 34

blockchain A blockchain is a linked list structure where nodes are called blocks. The head of a list is the latest entry and it references the previous block by the block's hash.. 1, 44–47

coin A coin refers to an unspent transaction output. Each of these outputs has an address and therefore addresses may have one or more coins.. 45

decentralized vaults Vaults in the cryptocurrency domain are a special mechanisms which make it harder to steal coins. Typically these are implemented by third-parties in the form of coins that require multiple signatures to redeem. Decentralized vaults do not require third-parties to maintain safety but are supported by the algorithms and protocol. Merit provides mechanisms to create vaults that do not require third-parities.. 1

Ethereum A distributed computation platform and cryptocurrency that has turing complete smart contracts and a state storage mechanism.. 3

genesis address Is an address of the first beacon in the genesis block.. 6

genesis block The first block in the blockchain. It doesn't reference any prior block.. 45

input Each transaction has one or more inputs. An input contains a reference to a previous output by the transaction id and index. It also has a script which only contains data. This data is fed into the previous output's script for validation.. 45, 46

invite An invite is a special kind of transaction which, when sent to an address a beacon advertises, will activate that address. Without at least one invite, an address is considered deactivated or invalid and cannot send or recieve merit.. 8, 44

inviter address Is the address that a beacon references as the one that they want to be invited by.. 7

Litecoin A cryptocurrency based on Bitcoin which provides a better transaction rate and lower fees.. 3

merit Is the name of the coin used in the Merit project.. 45

opcode An opcode is a command that is executed in a script by the Merit virtual machine.. 20, 46

output Each transaction has one or more outputs. And ouput contains the amount of merit and a script. An output is also called a coin and is considered spent when another transaction's input references the output successfully. An output is unspent when there are no valid transactions that reference it.. 46

proof-of-work Proof of work is some data that proves a machine did a given amount of work. Bitcoin uses a proof-of-work called hashcash. Merit uses a proof-of-work called the Cuckoo Cycle [2].. 39, 46

public key A public key is a set of bytes derived from a private key. It is used to decrypt data encrypted with a private key. It is also used to validate signature made by a private key. The public key can be shared freely with anyone or even published publically, hence the name.. 5, 44

regex A regex is short for *regular expression*. A regex defines a search pattern using a special language. A piece of text either matches or doesn't match the regex.. 9

script A script is composed of operations and data. Each output has a script which executes the transaction validation. Merit iherits the virtual machine from Bitcoin and extends it with it's own opcodes. 45, 46

shard A shard describes a region in a problem space. Shards do not overlap.. 46

sharding Breaking up a problem space into shards. 34

transaction Is an entry in the blockchain ledger. It contains inputs which transfer Merit to the outputs.. 1–3, 44–46

turing complete Turing complete means that a particular machine can do anything a Turing Machine can do.. 46

Turing Machine Is a mathematical model of a computer invtented by Alan Turing. It is a machine that can perform any computation.. 46

vault A vault is a special script that keeps Merit safe. It usually splits the functionality into spending and changing a vault. Spending requires a special spend key and changing requires a master key. Spending is restricted in several ways such as only being able to send to a whitelist of addresses, and also having a spending limit per transaction.. 2

virtual machine Is an implementation of a computer in software. Merit has a virtual machine to execute scripts. Merit's virtual machine is not turing complete to simplify reasoning of scripts and improve safety.. 45

warp warps are a blockchain that doesn't require a proof-of-work, but only a signature from a leader. Who get's to write is determined by the weave. Each block in the warp has transactions and is signed by leader.. 39, 44, 47

warp block A block that contains only transactions. For any given warp a single writer can write blocks. The writer is called the leader. The warp block does not require a proof-of-work but does require a signature from the leader.. 36

weave A weave is composed of weave blocks. A weave is analogous to the original blockchain instead of combining transactions into a block, it contains a claim by the miner to write into a warp. It also contains all coinbase transactions and invites.. 44, 46

weave block A block that contains only coinbase transactions, coinbase invits and claims to write to a warp. 36, 39, 47

References

[1] Aron Laszka, Benjamin Johnson, and Jens Grossklags, *When Bitcoin Mining Pools Run Dry*, Financial Cryptography and Data Security: FC 2015 International Workshops, January 30, 2015, Revised Selected Papers (pp.63-77)

[2] Tromp, *Cuckoo Cycle*, https://github.com/tromp/cuckoo/

[3] F. C. Kingman, *Poisson Processes*, (17 December 1992) Clarendon Press, ISBN 978-0-19-159124-2

[4] Pranav Gokhale, *Why is the Poisson distribution relevant to Bitcoin/blockchain?*, https://www.quora.com/Why-is-the-Poisson-distribution-relevant-to-Bitcoin-blockchain, Feb 21, 2017

[5] Antoon Bosselaers, and Bart Preneel, *The hash function RIPEMD-160*, https://homes.esat.kuleuven.be/ bosselae/ripemd160.html

[6] P. Faltstrom, Ed., *The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)*, https://tools.ietf.org/html/rfc5892, August 2010 ISSN: 2070-1721

[7] Blanchet Viagenie, *Preparation of Internationalized Strings ("stringprep")*, https://tools.ietf.org/html/rfc3454, December 2002

[8] Evgeniy Gabrilovich Alex Gontmakher, *The Homograph Attack*, http://www.cs.technion.ac.il/ gabr/papers/homograph_full.pdf, Communications of the ACM, 45(2):128, February 2002

[9] Mark Maunder, *Chrome and Firefox Phishing Attack Uses Domains Identical to Known Safe Sites*, https://www.wordfence.com/blog/2017/04/chrome-firefox-unicode-phishing/, April 14, 2017

[10] Jean-Philippe Aumasson and Daniel J. Bernstein, *SipHash: a fast short-input PRF*, https://131002.net/siphash/siphash.pdf

[11] Luc Devroye, *Non-Uniform Random Variate Generation*, http://www.eirene.de/Devroye.pdf, New York: Springer-Verlag. 1986

[12] *Script*, https://en.bitcoin.it/wiki/Script

[13] E. Rescorla, Ed., *Internet Denial-of-Service Considerations*, https://tools.ietf.org/html/rfc4732, November 2006

[14] Efraimidis and Spirakis, *Weighted random sampling with a reservoir*, https://doi.org/10.1016/j.ipl.2005.11.003, Information Processing Letters, Volume 97, Issue 5, 16 March 2006, Pages 181-185

[15] Horst Feistel, *Cryptography and Computer Privacy*, http://www.apprendre-en-ligne.net/crypto/bibliotheque/feistel/index.html, Scientific American, May 1973, Volume 228, No 5, pp. 15-23

[16] Alyssa Hertig, *Bitcoin Fees Are Down Big: Why It Happened And What It Means*, https://www.coindesk.com/bitcoin-low-fees-why-happening-why-matters/, Feb 23, 2018, at 01:30 UTC

[17] Peter R. Rizun, *A Transaction Fee Market Exists Without a Block Size Limit*, https://www.bitcoinunlimited.info/resources/feemarket.pdf

[18] Coindesk *How Will Ethereum Scale?*, https://www.coindesk.com/information/will-ethereum-scale/

[19] Wikipedia, *Warp and weft.jpg*, https://en.wikipedia.org/wiki/File:Warp_and_weft.jpg

[20] Ittay Eyal, Adem Efe Gence r, Emin Gün Sire r, and Robbert van Renesse, Cornell University, *Bitcoin-NG: A Scalable Blockchain Protocol*, https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf

[21] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*, http://www.rfc-base.org/txt/rfc-2396.txt

[22] By Horst Feistel, *Cryptography and Computer Privacy*, http://www.apprendre-en-ligne.net/crypto/bibliotheque/feistel/index.html

[23] Sergey Brin, Lawrence Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, http://ilpubs.stanford.edu:8090/361/1/1998-8.pdf

[24] John (JD) Douceur, Thomas Moscibroda , *Lottery Trees: Motivational Deployment of Networked Systems*, SIGCOMM 2007: ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan