OPAL: An Organic Protocol for Autonomous Ledgers

Thomas R. Stovall
trstovall@gmail.com
http://trstovall.github.io/

December 6, 2014

Abstract

Bitcoin has proven that a digital monetary system can operate without a central authority. However, the high price volatility and the deflationary aspects of the currency stall widespread adoption. This paper describes a new approach to decentralized digital currency. By localizing consistency constraints, incentivizing partition resistance, and adding voting capabilities, Opal seeks to create a decentralized digital currency with high network availability which smoothly matches currency supply to demand.

1 Introduction

Witnessing the havoc wreaked by the collapse of the 2007 housing bubble draws one to question the efficacy of the world monetary system as enforced by central institutions. While diluting the value of money discourages hoarding of currency, thereby pushing capital into growing markets, it also strengthens market bubbles. Taken individually, such market bubbles can be benign. But when combined, like with superposition of sound waves, the results can be devastating. As no real policy change has been the consequence of the recent economic collapse, it would seem that the superficial lust for job creation by policy makers outweighs the more fundamental desire for sound money. Hence, to build a more robust monetary system, control over monetary policy must be reclaimed by individuals seeking a decentralized solution. Such a solution may take the form of cryptocurrencies.

A cryptography-based digital currency (or cryptocurrency) is system where an identity in a payment network is publicly recognized as owning some portion of the total money supply; however, the individual corresponding to the identity may not be known. This is because an identity is a pseudonymous key in a public key infrastructure. Furthermore, an individual may have many identities. Currency is transferred when one identity digitally signs a message authorizing the network to reassign ownership of an amount of currency to a different identity, and the message is accepted by the payment network. Note that only the authority over a sum of money has been transferred. The signed message simply informs nodes in the network that the ledger has changed.

Any currency needs a policy governing the rate of monetary inflation and deflation. In Opal, the policy is decided through voting. Voting is performed through spending the currency and is weighted proportional to the amount spent. Spending the currency destroys any previous vote associated with it and creates a corresponding vote allowance which allows the spender or the network to influence consensus. Through consensus, the network has granular control over the inflation and deflation of the currency, the velocity of the money supply, the rate at which the ledger grows, and future amendments to the protocol. Furthermore, Opal uses a simple proof of work template which does not require the stamp to depend on previous stamps. This creates a market for miners to sell proof of work to different networks or the same network multiple times, enabling greater flexibility for the monetary and security requirements

of the network.

Imagine a small community which decided to move to a purely spoken monetary system – no coins, clam shells, or bank notes. When any two members of the community meet, they describe all payments which have occurred since they last met. A member only accepts payments as authentic when he or she has discussed it face-to-face with the payment sender. This prevents payment forgeries. The payment receiver can then spend the received funds only after a majority of its peers has authenticated the payment. This prevents counterfeiting, or double spending, the received funds. (Clearly, there are incentives to be sociable.) Hence, the community has a payment system, but from where does the money come?

The community has agreed that the money supply should increase at a given rate. At any point in time, each member knows what the current money supply is and should be. So, the calculated difference between what is and what ought to be the money supply is available to be freely claimed. Following the rules of the payment system (as the claims are simply payments), the available balance is portioned among the community members.

While this conceptual monetary system may be cumbersome (and creepy) in social networks, it can be fast, decentralized, and anonymous when operated on computer networks. Opal builds on the concept as a foundation, extending it with cryptography for privacy and scalability, digital networking for fast, global communication, and voting for an evolving, decentralized monetary policy and protocol. This paper describes the conceptual elements of the Opal protocol and some suggested implementation details. The rest of this paper is organized as follows: the next section discusses some related or influential work and competing alternatives. Then in section 3, the monetary policy of Opal is compared to that of the leading cryptocurrency. Section 4 describes the signature scheme and how a router in the network can get paid for its work without needing to be a miner. Sections 5 through 7 discuss how payments are represented and communicated to the network. Mining is then discussed in section 8, and network ledger consensus in section 9. Section 10 explores stakeholder voting, followed by a discussion of Opal's mechanisms for securing privacy in section 11.

2 Related Work

As an anti-spam and denial of service countermeasure mechanism, Adam Back introduced Hashcash [1] proof of work in 1997. Proof of work is the repeated computation of random inputs, to generate an output string which satisfies a well-known challenge requiring some expenditure of computational resources. The input which corresponds to an output satisfying the challenge is known as a proof of work stamp. A unique stamp would then be attached to a web resource request to grant the request allowance of some resource. The Opal network requires that each transaction, a set of atomic changes to the ledger, include a proof of work stamp representing a sufficiently large amount of work.

Inspired by Back's work on Hashcash, in 2008, Satoshi Nakamoto published a paper detailing a way to use proof of work for securing payments and deterministically inflating the money supply of a digital currency network called Bitcoin [8]. The network relies on competing nodes to publish proof of work stamps which represent a sufficiently large amount of computational effort to claim the periodic block reward, a combination of inflationary seigniorage and payment processing tips. The proofs of work nodes, called miners, are also expected to update the ledger with the latest valid payments. This model could be seen as a single ledger shared by many nodes, with the many nodes competing to publish payments. Opal relaxes these consistency constraints encouraging each node to maintain its own autonomous ledger while convincing all other nodes to accept the node's ledger updates.

Besides allowing simple payments, Bitcoin includes a scripting language which allows for a payment to be accepted depending on a complex set of conditions. This creates the possibility for automated conditional payments, or smart contracts. The Ethereum [4, 10] network implements a currency with a Turing complete scripting language, allowing payments which are determined by arbitrarily complex contracts. The Cryptonite [3] currency takes a radically different ap-

proach, omitting a scripting language entirely. Opal follows the Cryptonite approach; while the transaction data structure is flexible, and consensus governs the rules and parameters of the protocol, Opal has no scripting language. Because the proof of work stamps used in Opal can be reused in many networks, it is sufficient to have each network implement a few specialized contracts, rather than one network supporting many arbitrarily complex contracts.

Ripple [6] is a decentralized credit based payment network and web of trust. The Ripple network seeks to minimize the trust required to accomplish consensus by requiring each node to trust a variety of conflicting organizations. The driving idea is that opposing nodes are unlikely to collude. Ripple nodes then proceed through a series of voting rounds until each peer comes to consensus with its trusted peers on which ledger changes are to be accepted. This voting system localizes ledger consistency requirements, allowing the network to come to consensus much faster than an authority periodically elected by proof of work. Opal adopts this decentralization of consensus, but attempts to maintain node anonymity by forming the web of trust through quid pro quo communication.

3 Monetary Policy

In a monetary system, currency is issued by the authority of that system as a symbol of debt to individuals who perform some task for the authority. Later, the currency is partially or entirely returned in reconciliation of a debt to the authority, i.e. taxes or transaction fees. In national systems, this authority is a centralized institution. In precious metal based systems, it is the miners. In Bitcoin and Opal, it is the payment network.

The most popular monetary policy in cryptocurrencies, namely that of Bitcoin, is the deterministic, decreasing rate of monetary inflation until a specified supply is reached. Although this policy is algorithmically simple, demand shocks cannot be absorbed and are expressed as price volatility. Bitcoin enthusiasts argue that the demand shocks are not absorbed because the currency is still too small. While this may be true, it is unlikely that bitcoin will ever be able to scale to sufficient dominance without either changing its transaction validation algorithm or reducing the number of validators. Another sentiment is that the price volatility is rewarded by an overall increase in price. However, there are many extended periods that serve as counterexamples to that sentiment. Other currencies, like Feathercoin, have not been as successful as Bitcoin and illustrate this point even better.

Ultimately, it is not the price of the currency that matters, but the change in price. Opal gives the network control over the monetary supply. This means that changes in demand can be met with changes in supply, and the currency can attain a normative value. The normative value can then be enforced by the network's willingness to buy proof of work, and miners' willingness to sell opal. That is, currency can be bought or sold in reaction to small demand shocks. For larger changes in demand, there will be a reserve of work available. In periods of decreasing demand, the network will pay less for more work, miners will build a reserve, and the monetary inflation rate will decrease. Conversely, in periods of increasing demand, the network will pay more for less work, miners will unload their reserves, and the monetary inflation rate will increase. Ideally, this will allow Opal to scale down to serve many small communities independently, without suffering from high volatility.

4 Signatures and Signed Sets

Decentralized payment networks need a way to pay for their infrastructure. Without the monetary incentive to provide resources to the network, running the network becomes an overhead cost, and the good will of the participants erodes. Since centralized institutions are able to amortize costs over time and clients, without proper incentives, a currency may become centralized.

Bitcoin seeks to solve this problem by using the distribution of currency to subsidize miners, where mining acts as the security infrastructure. However, it is not enough. The subsidies buy a high degree of consistency, but availability and partition resistance

are left as expenses. Combining this with the increasing number of transactions shows a network that is being pushed toward a single server model, where availability and partition resistance are less costly. Hence, cryptocurrencies need a means for incentivizing the part of the system that connects miners and merchants together – the network.

Consider a sender with a balance of 10 opal. If the sender sends 3 opal to each of three recipients, then 1 opal remains as a tip. In Bitcoin, only the miner which processes the payment can collect the tip. By allowing messages to be added to a signed set of messages, routers in the Opal network can collect (portions of) the tip.

A typical EdDSA signature [2] is (R, S), where $S = r + H(A, R, M) * a \pmod{l}$, for session key $R = r * B$, public key $A = a * B$, cryptographic hash function H, message M, and the elliptic curve's base point B. The signature (R, S) validates message M under public key A if and only if $S * B = R + H(A, R, M) * A$. Opal signatures take the modified form $S = a + H(M) * r \pmod{l}$, and verified iff $S * B = A + H(M) * R$. The Opal form is suggested in the EdDSA paper as an equivalent alternative, where A is omitted from the hash input to decouple payer from payee, and R has been omitted to optimize multiparty signatures. While the omission of R may weaken the security of the signature scheme for general application, the scheme remains practically secure for Opal.

Opal (and EdDSA) signatures can be combined for batch verification through addition, i.e. for n signatures, $S = \sum_{i=1}^{n} S_i \pmod{l} = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} H(M_i) * r_i \pmod{l}$. Furthermore, if S is not yet public, then the signer can publish $S_{n+1} = (S_n + a_{n+1}) \pmod{l}$ as the signature under an additional key. Similarly, additional messages may be added to the set without breaking the signature. Generalizing this for one or more messages signed under one or more public keys, the signature $S = \sum_{i=1}^{n}(a_i) + \sum_{j=1}^{m}(H(M_j) * r_j) \pmod{l}$ is valid if and only if the equation $S * B = \sum_{i=1}^{n}(A_i) + \sum_{j=1}^{m}(H(M_j) * R_j)$ holds.

In EdDSA, $R = r * B$ where $r = H(a, M)$. This derivation was chosen so that R is unique for each signed message and does not depend on the security of a random number generator. In Opal, R must be unique for each set of signed messages. Otherwise, the standard approach may leak private keys if, for example, a signer signs messages M_1 and M_2 together under public key A, then each individually under A. Trivially, $S_1 + S_2 - S_{12} = a$. Thus, generalizing the signature scheme requires generalizing the derivation of the session key R. To derive R deterministically for a signed set of m messages under n public keys, r should be computed as $r = H(a_1, ..., a_n, M_1, ..., M_m)$. Opal goes a step further and uses unique session keys for each message in the signed message set. That is, each message M_j has a corresponding session key R_j such that $R_j = r_j * B = H(H(M_j), r) * B$.

5 Merkle Trie

A Merkle tree is a tree in which every non-leaf node is labelled with the hash of the labels of its children nodes. Hash trees are useful because they allow efficient and secure verification of the contents of larger data structures ("Merkle Tree", *Wikipedia: The Free Encyclopedia.*). Unfortunately, the root hash of the tree depends on the order in which elements are added to the data structure. In a decentralized network, where elements are created concurrently, a Merkle tree can place unrealistic demands on network consistency.

Opal makes extensive use of Merkle tries as well as Merkle trees. An array mapped trie is a space efficient search tree where the keys of the map are encoded as the paths in the tree. Consider a Merkle tree which represents a SHA3-512 hash array. (The length of such a hash array would be 2^{512} elements.) Let each leaf node n of the Merkle tree have the label $L_n = H(n)$ and each branch node br have the label $L_{br} = H(L_a, L_b)$ where a and b are children of br. Also, let $\emptyset$ represent labels or nodes which do not exist, $H(x) = \emptyset$ if $x = \emptyset$, $H(x, y) = H(y, x) = H(y)$ if $x = \emptyset$, and $H(x, y) = H(x|y)$ if $x, y \neq \emptyset$, where H is the a cryptographic hash function and $|$ denotes concatenation. Then, this Merkle tree can be efficiently represented as a hash array mapped trie, and the labels of branch nodes in the tree (specifically the root

label) do not depend on the order in which the nodes' children were added. Hence, the Merkle trie is a set rather than a list, but with the security properties of a Merkle tree.

6 Payments

A payment is a signed set of messages where at least one of the messages authorizes the spend of a sum of funds. In Opal, this is either a proof of work stamp, or an input credit. To validate a payment, two payment accounts are created, one for transferring funds and one for voting. Then, the value of the work stamps and/or credits in the payment is credited to both accounts. A spend authorizes some amount of funds to be credited to a later payment, and that amount is debited from the transfer account. Votes debit the voting account. A vote must be associated with a spend, and the spend amount is the amount debited from the voting account. The network applies fees to the payment according to the size, in bytes, of the payment, which is debited from both accounts. If both resulting account balances are non-negative, and the signature is valid, then the payment is valid.

This style of payment allows for various types of payments. The simplest payment message is a balance transfer, such as: ((opal/in, spendable), (opal/out, spend, msg_key), (signature, S)). Alternatively, a miner may request a payment for work, combine the payment with a previous sum of funds, and associate the result with a vote to change the protocol: ((opal/work, stamp), (opal/in, spendable), (opal/out, spend, msg_key), (opal/votes, vote, msg_key) (signature: S)). Lastly, an individual may simply want to pay a fee to publish some data, {*'abc':'xyz'*}, to the network under an arbitrary search key, *name*: ((opal/in, spendable), (name/abc, xyz, msg_key), (signature: S)).

7 Transactions

A transaction is an atomic set of proposed changes to be applied to the ledger; the changes must uniformly succeed or fail. The following pseudo-structure illustrates the anatomy of a transaction:

```
version: [protocol_hash, msg_key1]
references:
  in:  [[tx_hash1, msg_key2], [tx_hash2, msg_key3], ...]
  out: [[tx_hash2, msg_key4], ...]
opal:
  work: [stamp1, stamp2, ...]
  in: [spendable1, spendable2, ...]
  out: [[spend1, msg_key5], [spend2, msg_key6], ...]
  votes: [[vote1, msg_key7], [vote2, msg_key8], ...]
signature: S
```

This structure is then decomposed into the following set of elements and hashed as a Merkle trie:

```
1:  (version        | protocol_hash | msg_key1)
2:  (references/in  | tx_hash1      | msg_key2)
3:  (references/in  | tx_hash2      | msg_key3)
4:  (references/out | tx_hash3      | msg_key4)
5:  (opal/out       | spend1        | msg_key5)
6:  (opal/out       | spend2        | msg_key6)
7:  (opal/votes     | vote1         | msg_key7)
8:  (opal/votes     | vote2         | msg_key8)
9:  (opal/work      | stamp1                  )
10: (opal/work      | stamp2                  )
11: (opal/in        | spendable1              )
12: (opal/in        | spendable2              )
13: (signature      | S                       )
```

The contents of a transaction are organized into a tree of tagged elements. Like with HTML, the tags indicate how the element is expected to be processed. For transaction validation, the version, references, signature, and at least one work element are required. All others are optional. Elements of the transaction are then serialized and inserted as leaf nodes into a new Merkle trie, and the root hash of the trie is used to identify the transaction.

- *version* is the description of the protocol outlining the set of rules for interpreting the messages in the transaction. The described protocol must be compatible with the protocol as determined by the votes in the referenced transactions. The protocol is described by *protocol_hash*, the root hash of the Merkle trie containing the set of rules

and network parameters required to interpret the transaction. To ensure the version message is included in the signed transaction, a signature session key, *msg_key*, is used to lock it into the transaction.

- *references* is a minimal set of non-conflicting transactions which describe the prior state of the global ledger. Two transactions conflict if they share an input spendable. A transaction is included in the prior ledger of this transaction if its transaction hash appears in the *references/in* set of this transaction (or the corresponding *references/in* set of an included transaction) until it is excluded from the ledger by appearing in the *references/out* set of a transaction in the ledger. Hence, each transaction describes a new global ledger. A peer can then use this set of references to determine account balances and validate the transaction *version*.

- *opal/work* describes a set of proof of work stamps. Each stamp is a concatenation of *type*, *public_key*, *bits*, *zbytes*, and *nonce*. *type* indicates the hash algorithm and format of the stamp, allowing the protocol to use multiple hash functions for proof of work. *public_key* is a compact representation of the miner's public key and is included into the transaction's list of public keys. *bits* is an integer describing the amount of work. *zbytes* is an integer indicating the number of zero bytes to left pad the *nonce*, and *nonce* is a byte string of random data. Let $h = SHA256d(stamp)$, where $SHA256d$ is two rounds of the SHA-256 hash algorithm. If h has at least *bits* trailing zero bits, *bits* is in an acceptable proof of work range, and the stamp has not been used too recently according to the protocol, then the stamp is accepted, and the transaction's spend and vote accounts are credited with the appropriate subsidy for the proof of work. Otherwise, the stamp (and the transaction containing it) is rejected.

- *opal/in* is the set of spendables to which previous spends were paid. A spendable is a nested multisignature claiming previous outputs. For example, let $spendable := (type, (1, (1, pk, kh1), kh2))$, where pk is a revealed public key, and *kh1* and *kh2* each are truncated hashes of public keys or truncated root hashes of multisignatures. Then, spendable is a 1 of 2 signature nested inside of another 1 of 2 signature, and the public key, *pk*, is included into the transaction's list of verification keys. A spendable claims the output balance of any account which is a prefix of the hash of the spendable. Note that the spendable is constructed in such a way so that the root hash of the spendable does not depend on which public keys are revealed for signing.

- *opal/out* is the set of spends enumerating the accounts for which to create balances, along with the balance amounts. An *account* is the truncated root hash of a spendable to associate with a *balance*. Each *balance* is debited from the transaction's spend balance and credited to the account's balance. Again, any valid input spendable with the spend *account* as a prefix spends the account's balance. The balance amount is formatted as a list of signed bytes, where each byte represents a power of two denomination of opal.

- *opal/votes* are proposed changes to the protocol, and are associated to unspent outputs. Votes are weighted according to the amount in the output's account. Loose coupling of votes to outputs allows unclaimed votes to be claimed by anyone in the network. Voting is later discussed in more detail.

- *signature* is the modified EdDSA signature of the transaction and is simply the sum of the signatures of the payments included in the transaction.

8 Mining

In the domain of cryptocurrencies, mining is the task of polling random numbers from a random oracle until an element of the agreed upon subset of possible

numbers is found. That is, given a random oracle $RO : M \rightarrow N$, miners are looking for an input m, known as a *proof of work stamp*, such that for output $w = RO(m)$, such that $m \in M$, $w \in W$, and $W \subset N$. Hence, if the set W is much smaller than the set N, then finding the desired input represents a large expenditure of effort.

A typical exchange on the web consists of a client sending a request to a server. The server then responds by sending back a resource. The resource may be a web page, a media file, an email, etc. Consider a web server which can handle a maximum of 1,000 requests at any given time. If each client sends only one request, then the server can handle 1,000 clients concurrently. But if a dishonest client makes 1,000,000 requests at once and the server responds to random requests, then the honest clients have a low probability of receiving the requested resource. A typical solution to resolve this problem is to identify the dishonest client and ignore requests from it. However, a sophisticated attacker can mask its identity. An alternative solution would be to require every client to send as many requests as it can until it receives the requested resource.

An optimization to the second solution would be to combine many requests into a single request and send the combined request to the server. Suppose that the server hashes each request and ignores requests whose hashes do not have a minimum number of leading zero bits. If the server publicly shares the minimum number of leading zero bits required, clients can generate requests and hash them until a request is found such that the hash of the request meets the server's requirement. The client then sends the request. This approach, known as *proof of work*, limits the number of requests a client can make and minimizes congestion in the network.

Communication in decentralized digital currencies reflects the web's idea of request-response, except the resource is a currency transfer acknowledgement, the server is a collection of peers in an open network, and parameters like the minimum number of zero bits are decided collectively by the peers in the network. Because the work required to make a request may be impractical for some clients, they can send a request to a peer specializing in generating requests on behalf of others in exchange for tips and currency seigniorage.

In Opal, each transaction must include a minimum amount of work. The network should set the rules of the proof of work algorithm such that the probability of finding a proof of work stamp leads to low variance for average miners, without flooding the network with proofs of work. Opal also relaxes the rules so that a proof of work stamp does not depend on a previous proof of work, and multiple hash functions may be used for proof of work mining.

Proof of work stamps may be redeemed for currency multiple times, given that they continue to meet the rules of the proof of work system, and enough proof of work has been redeemed since the stamp was previously redeemed. This provides a few benefits. Miners have a running stake in the success of the network, and so may be less likely to flood the market with cheap money in times of panic. Also, multiple redemption of proof of work should act as a buffer against attackers who horde proof of work then flood the network. Lastly, a transaction is validated against the rules it references (described in section on Rules and Voting); so, reused proofs of work are not likely to ignore new ledger, policy and protocol changes.

It may be unclear why there should be mining at all. One purpose of mining is to seed the network with authority. If transactions must contain proof of work, peers who report the most valid, unique transactions are likely to be the most honest, well connected peers. Mining also allows other aspects of the network to be regulated, such as transaction creation rate, transaction size, ledger length, etc.

9 Consensus

After connecting, two peers begin inter-node consensus with an empty shared ledger. Each peer then sends to other peer the set of accepted transaction items (transactions or transaction fragments) which have not yet been included in the shared ledger. Typically, this set should be small, as transactions reference previous transactions and accepting a transaction implies that all referenced transactions have

been accepted. To accept a transaction item, a peer issues a *put* command for the item's hash. If the second peer does not know about the item, it will issue a *get* command for the item's hash and listen for the first peer to send the corresponding item. If and when the second peer accepts the item, it too will issue a *put* command on the item's hash. A peer can then remove a transaction or set of transactions from the shared ledger by accepting a transaction or fragment which conflicts with the previously accepted item. This process continues until the peers close the connection.

Intra-node consensus is similar, but stricter. Instead of accepting all valid transaction items, a node accepts only transactions which appear in a majority of its peers' shared ledgers. This intranode ledger represents the set of confirmed transactions, and it is from this set that a transaction forger will pull transactions to reference, a client will derive its account balances, or a new peer will begin with its initial ledger. The node's ledger does not include transaction fragments. Instead, they are maintained in a separate set. Peers then pull from this set to share the fragments with other peers.

Given this form of exchange, a node builds an accurate representation of the true global ledger by maximizing its communication with the network. Although it is up to each node to determine how to achieve this, an obvious approach would involve evaluating each connection according to profitability, health, reliability, and cost.

Because any node can contribute to a transaction as it is being routed through the network, routers in the network may add outputs, votes, notes or any other valuable message to the transaction. However, modifying a transaction resets its visibility by the network. Hence, a router should likely only add *tips* to a transaction sparingly and only when it believes the transaction has gained little visibility since creation. Nodes which effectively propagate the modified transaction should be seen as profitable.

Each transaction must include a network determined amount of proof of work. As this is obtained through provably occupying mining hardware, a limited resource, nodes which are first to submit eventually accepted transactions to the network should be seen as healthy. Other factors may go into evaluating health, such as the accuracy of time as reported by new transactions.

As consensus is the goal, a node's reliability can be measured based on the degree to which it agrees with other nodes. Also, nodes with longer connection times may be considered more reliable.

Finally, the benefits provided by a peer should be normalized by the cost of maintaining the connection. A peer node may often be the first to relay a new transaction, but it may also be relaying many conflicting or invalid transactions. So, these costs should also be considered.

10 Voting

Since each node independently decides which transactions are valid, it is not entirely necessary for a system like Opal to have a formal set of rules for voting. However, voting binds together the participants in the network. Voting gives a predictable set of rules for transactions to be accepted. It helps the network to move as one in consensus. Most importantly, voting allows the network to express the will of those holding the currency as a compliment to the will of the transaction forgers that govern the network.

There are two categories of issues to be decided in the Opal network. The first covers the features of the network. Features determine what can be done in the network. That is, features govern how transactions are formatted and validated. To add or remove features requires a super-majority, initially set at 70%. As new features typically require software changes, the super-majority requirement imposes a conservative rate at which the software evolves. This should encourage a diversity of developers and implementations, but may lead to the protocol becoming ossified. The second category covers parameters. Parameters determine how and to what degree features in the protocol operate. Examples of parameters include the transaction forging subsidy amount, the transaction data fee, and the difficulty range for valid proof of work. Modifying parameters places a much lighter burden on protocol implementors than modifying features. Also, the state of the parame-

ters in the protocol typically represent a balance of self-interest among the various factions in the network. For this reason, modifying parameters requires a simple majority of 50%.

Voting is an option made available through spending. When an individual spends opal, any vote associated to those funds is destroyed. The spender then has the opportunity to add a message to the payment, associating a vote to an unspent output in the payment. The weight of the vote is the amount transferred to the output, and each output can have a maximum of one vote referencing it. The vote is then counted and applied to the protocol when the payment is confirmed in a valid transaction. The vote is expressed in the payment by the root hash of the Merkle trie defining the protocol as it should be.

Votes are counted according to a weighted percentile. Consider a parameter represented by a variable with more than two states. Votes are ordered in a sequence according to the votes' proposed value for the variable. Unclaimed votes are inserted into the sequence proposing the most recent pre-vote value of the variable. Starting with the least proposed value, the sequence is scanned, and the weights are accumulated until the accumulated value is greater than or equal to the target value, i.e. the necessary percentage of the total available funds. The most recently scanned vote's proposed value becomes the suggested value. If the accumulated value is greater than the target, then the variable is set to the suggested value. If instead the accumulated value equals the target value, one of three branches is taken. First, if the suggested value is equal to the pre-vote value then no change is applied. Second, if the suggested value is less than the pre-vote value, then the variable is set to the proposed value of the vote next in the sequence. Third, if the suggested value is greater than the pre-vote value, then the proposed value of the previous vote in the sequence becomes the variable's value. Trivially, for a variable with two states (like a feature being enabled), if the pre-vote value has at least 100% minus the target majority, then no change is made; otherwise, the variable changes states.

The following are examples of parameters in the protocol:

- *Transaction subsidy* The subsidy is paid to the forger of a transaction and scales with the effort required to produce the proof of work included in the transaction. Because this parameter controls the monetary inflation of the network, the value of this parameter is a compromise between the network infrastructure providers (miners and transaction forgers), and the holders of the currency.

- *Transaction history length* Once some amount of proof of work is claimed since a transaction is accepted, the transaction must be removed from the ledger. Any unspent outputs in such a transaction are forgotten by the network. So, decreasing the history length lowers the burden on the infrastructure providers, but it also decreases the ability of the currency to be used as a store of value.

- *Proof of work period* The period determines how much proof of work must be claimed before a proof of work stamp may be used again. Decreasing the period increases the rate at which new transactions are formed. This is better for fast payments, but leads to smaller tips for transaction forgers. Notice that this parameter has a practical upper bound equal to the transaction history length.

- *Proof of work range* This parameter, actually two parameters, places an upper and lower bound on the expenditure of effort required to produce a proof of work stamp. Increasing these values together increases the profit miners make from publishing stamps, but also increases the period of the creation of new stamps. While big miners may see an advantage to increasing both values, mining could become impractical for small miners. Decreasing the values together increases the creation rate of new stamps which may flood the network. Narrowing the range increases the uncertainty for mining return on investment, while broadening the range could stratify the network into large, centralized miners and small, spammy miners.

- *Transaction data fee* The fee places a burden on spam and counteracts the monetary inflation of the transaction subsidy by removing currency from the supply. Plainly, a high data fee increases the costs of payments, while a low data fee encourages network spam and price inflation. Both scenarios could trigger capital flight and devaluation of the currency.

Notice that these parameters are coupled by various real world incentives. For instance, decreasing the proof of work period without decreasing the transaction subsidy will increase the rate of monetary inflation. Increasing the transaction history length without increasing the transaction data fee will increase the burden on the network infrastructure. Hence, the set of parameters in the protocol should be taken as a whole.

Technically, voting is enabled at network launch. However, A large amount of opal is initially created, then immediately destroyed. Such is done to lock the protocol until enough proof of work has been submitted to the network and the lock transaction is forgotten by the network. This allows the network to bootstrap safely while otherwise destructive votes are buffered by honest, constructive voting.

11 Privacy

Privacy in a payment network is necessary for the survival of the network. Leaking an individual's identity could allow malicious actors to freeze the individuals ability to spend. This is equivalent to stealing the individual's funds and violating the individual's right to self-expression. Analyzing an individual's payment history expresses intention to violate an individual's right of free association. Both of these violations are a threat to the fungibility of the network's currency, which is fundamental to the health of the network.

Opal builds on the privacy model of Bitcoin. That is, payments are public agreements involving pseudonymous actors. However, payment pseudonymity alone is not enough to guarantee privacy. In Bitcoin, identity can be leaked in a number of ways. Some solutions have been developed to help prevent these leaks. Opal attempts to support such solutions natively.

Firstly, payments which leak identity, like point of sale, can be combined with social network analysis to link the leaked identity with other payments. This can eventually lead to the ability to build a full financial profile, which can mean targeted advertising, harassment, or persecution for individuals or business model analysis for companies. Coin mixing has been proposed and implemented as a remedy for this; however, because coin mixing currently takes an interactive approach, it is vulnerable to denial of service (DOS) attacks. Alternatively, one could interact with a service which provides DOS resistance. Unfortunately, this requires trusting the service not to leak identity information about its users.

Notice, Opal's signature scheme allows payments from different sources to be merged non-interactively. This provides a DOS resistant degree of obfuscation, but may weakly do so if the transaction includes very few payments. Also, the mixing service has merely been replaced by the Opal network, in which identity leakers may be participating. So, this is not a complete solution. However, this feature could be combined with parallel networks implementing CoinShuffle [9] or similar protocols to minimize payment traceability.

Also, an organization which has the ability to monitor all connections on the network could detect which node is the source for a specific payment before the payment gets mixed into a transaction. Since node identities in Opal are EdDSA public keys, communication between two nodes is easily encrypted using a key derived from a Diffie-Hellman key agreement. In a network where low latency is incentivized, encryption has clear disadvantages. Realistically, a node should not be expected to encrypt its communication entirely. It is enough for a node to maintain a certain percentage of connections which use encryption, and use those to broadcast payments originating from that node.

Encryption does not help when you are directly connecting to the identity leak. In Bitcoin, it is surprisingly inexpensive to directly connect to a large majority of the participating nodes in an effort to discover which payments originate from which nodes.

To mediate this, many users connect to the Bitcoin network through anonymizing services like Tor [5]. Since Opal nodes use public key identifiers, Tor could easily be co-opted into the network. Alternatively, since Opal nodes seek to maximize their influence in the network by maximizing number of connections, a node can achieve plausible deniability by publishing different payments to different subsets of peers. Combining this approach with anonymizing DC-nets like Herbivore [7], decentralized computing clouds, and coin shuffling services provides an opportunity for real privacy, not just confidentiality, in the network.

12 Conclusion

Opal has the potential to be an international decentralized currency like gold and silver. Through voting, monetary policy can be applied. Like cryptocurrencies, Opal provides a means of online exchange independent of a central institution, but can now scale to transaction rates of PayPal, Visa or Mastercard.

Like the United States Constitution, this protocol was designed to provide *checks and balances* by separating governing powers and incentivizing competition. Also, the protocol is highly extensible, with desired changes expressed through voting. However, (continuing the analogy) the protocol is left without a *Bill of Rights*. It has been left to those participating in Opal to provide anonymizing communication protocols for ensuring the freedoms of speech and association.

Allow one last analogy. The LORD created the sun, the moon, and the stars to represent the Father, the Son, and the Holy Spirit. The Father is the source of justice. The Son is the source of light in the darkness. The Holy Spirit is a helper to guide one to the Son and the Father.

13 References

References

[1] Adam Back. Hashcash, 1997.

[2] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Available at URL: http://ed25519.cr.yp.to/ed25519-20110926.pdf*, 2012.

[3] JD Bruce. Purely p2p crypto-currency with finite mini-blockchain. *Available at URL: http://diyhpl.us/~bryan/papers2/bitcoin/Purely%20P2P%20crypto-currency%20with%20finite%20mini-blockchain.pdf*, 2013.

[4] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. *Available at URL: https://www.ethereum.org/pdfs/EthereumWhitePaper.pdf*, 2014.

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. *Available at URL: http://pdos.csail.mit.edu/6.824-2011/papers/dingledine-tor.pdf*, 2004.

[6] Ryan Fugger. Money as ious in social trust networks & a proposal for a decentralized currency network protocol. *Available at URL: http://archive.ripple-project.org/decentralizedcurrency.pdf*, 2004.

[7] Sharad Goel, Mark Robson, Milo Polte, and Emin Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. *Available at URL: https://gnunet.org/sites/default/files/herbivore-tr.pdf*, 2003.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Available at URL: http://nakamotoinstitute.org/bitcoin/*, 2008.

[9] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. *Available at URL: https://www.petsymposium.org/2014/papers/Ruffing.pdf*, 2014.

[10] Gavin Wood. Ethereum: A secure decentralized generalized transaction ledger. *Available at URL: http://gavwood.com/paper.pdf*, 2014.