Hyper Poly Protocol
A Decentralized Communications System for Innovation

August 1th 2018

Abstract

We propose the Hyper Poly Protocol (HPP) Network, a communications platform, decentralized by means of creating a blockchain system. We achieve user privacy without the need for preliminary data exchange, using unlinkable transactions and ring signatures as default. The system helps to reduce spam and enables senders to bid for a user's attention. Several convenience features are included such as optional sender verification, efficient message retrieval and Twitter user verification, as well as an extendable protocol allowing for adapters, message compression and third-party recognized message headers. We've kept driving innovation at the centre of the network's design, allowing third party tools to be developed on top of the network with ease.

1 Introduction

Hyper Poly Protocol (HPP) works by extending a typical blockchain network to inject messages and attachments to transactions. These messages will naturally be encrypted in the same way as the rest of the transaction and various features are implemented to ensure end-to-end anonymity.

Other communications systems have been proposed, such as Bitmessage[1]. However, this system assumes the altruism of it's miners. This ignores the greatest reason for the stability and efficiency of blockchain protocols - the financial incentive given by the proof of work or proof of stake functions.

We believe using non-human-friendly hashes as an address will hamper the wider appeal of this network, as such, we are proposing various network adapters to verify users using different external means such as Twitter.

The network discourages from sending messages to an excessive number of recipients, to reduce spam, and allows the senders to bid for a user's attention.

2 Summary of features

- Sender anonymity and privacy
- Receiver anonymity and privacy

- Real-life user verification
- Receiver-only sender verification
- Attention bidding
- Efficient permanent message storage
- Adaptive difficulty
- Flexible soft-limit block size
- Message compression
- Message attachments
- Whitelisting senders for priority inboxes
- Message chaining
- Efficient method for recognizing a receiver's messages
- Features to assist third party innovation

3 Blocks and the HPP blockchain

3.1 Why a blockchain?

There are various reasons why the HPP Network has been chosen to be a blockchain technology over just a service:

- *Decentralization* - This removes the baseless trust in the good intentions of a service provider - be it in how they handle your data or how anonymous you truly are. It also removes the baseless trust required by the service on the underlying infrastructure provider.
- *Natural growth* - After the network is set up, there is natural incentive for miners to contribute and for third parties to develop on the network capabilities. As mining hashing power grows, by sheer allocated resources or improved equipment, so will transaction processing rates.
- *Open sourced* - No need to trust a closed source web service.
- *Free market* - Since message fees are not regulated, the going rate of transaction fees will be freely determined by the network users.

3.2 Building a new blockchain

A number of features of the HPP Network have drawn from and enhanced the work of other blockchains technologies, particularly Monero[2], though we emphasise the intention has never been to base the design on an existing blockchain technology, rather features of the network having naturally evolved to be based upon these.

The HPP Network will be a new blockchain rather than building upon an existing one. There are strong reasons not to build upon an existing blockchain:

- Ease to get new features adopted.
- Avoiding detrimental features being adopted.
- Transaction fees scaling appropriately.

3.2.1 Ease to get new features adopted

Take, for instance, storing the message contents off the blockchain in the message storage system proposed in this paper. It aims to reduce the impact large messages have on the blockchain size and is a mechanism that will help keep transaction fees low.

Whilst it is optional and the implementation details are up to the nodes, unless it is a widespread idea in underlying blockchain as well, large messages would cause the blockchain to be bloated for the majority.

3.2.2 Avoiding detrimental features being adopted

We also accept that an existing blockchain may adopt features that are a hinderance to the network, since coin transfer will still be the main purpose of the blockchain. As such, using an existing network would be a level of flexibility that would be sacrificed if the need arose.

### Transaction fee scaling

The transaction fees in blockchains tend to scale to how much users are willing to spend to transfer coins to one another. We foresee most transactions for the HPP Network would be to send messages to one another and that users would be willing to spend less to send a message than to transfer coins.

If we were to build the technology upon an existing blockchain, the majority of transactions on the blockchain would still be coin transfer transactions and so the fees would be scaled to reflect that, making it more expensive for users to send messages, which a lot simply wouldn't accept.

On top of this, message transactions are expected to be a significantly larger size than standard blockchain transactions, so miners would be expecting would be larger transaction fees than transactions for just transferring coins.

3.3 Blocks

The blocks are composed of three elements:

- *Block header* - Information used to join the blocks together and give the parameters for the proof of work.
- *Block hash* - A hash of the block header, used as an identifier of the block and as proof of work by the miner.
- *Transactions* - Transactions of the block.

3.3.1 Block headers

The block header structure is similar to that used by Bitcoin[3]:

Key	Usage
Version	Version number of the protocol, loosely following semantic versioning

Key	Usage
PreviousBlockHash	The hash of the header of the previous block
MerkleRoot	The Merkle tree of the transactions root hash
Timestamp	Timestamp of the block creation, seconds from Unix epoch
DifficultyTarget	Proof of work target difficulty
Nonce	Used as part of the proof of work algorithm

Example blocks headers structure

```
{
  "Version": "1.0.0",
  "PreviousBlockHash": "0000013687b43cd38c03c959ef3",
  "MerkleRoot": "8a106c86440545709a9e7816cdb0",
  "Timestamp": "1503150470",
  "DifficultyTarget": 2.100,
  "Nonce": "a83681de22db7ce16672f16f3"
}
```

3.3.2 Transactions

HPP transactions will initially take three forms which are discussed in detail in the following sections:

- *Coin transfer* - HPPCoin transfer from one address to another.
- *Message transfer* - Message transfer from one address to another.
- *Message retrieval transaction* - Long term message retrieval.

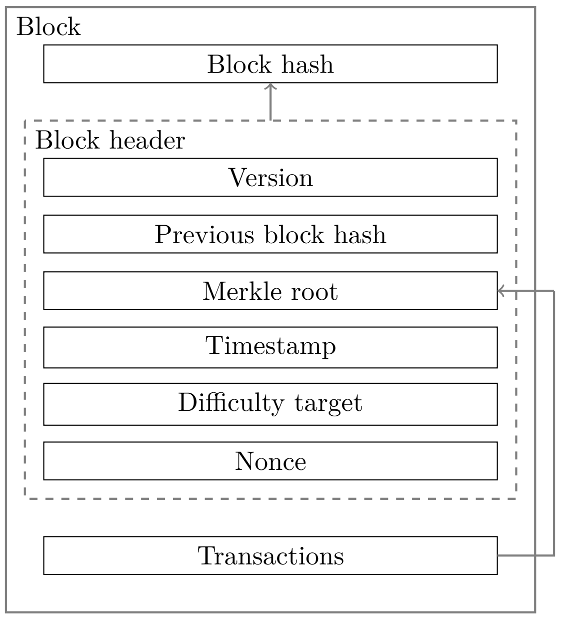


Figure 1: HPP block structure

4 Blockchain scaling

Since messages are delivered in the same transactions that are placed into blocks, the blockchain could grow at a very fast rate. We see in Bitcoin that the average transaction is about half a kilobyte, yet the blockchain is over 130 gigabytes as of the end of August 2017[4]. HPP transactions will be larger than Bitcoin transactions on average due to:

- Containing the messages.
- Containing the message headers.
- One-time ring signatures instead of a single sender signature.
- One-time public key, being twice the size of a Bitcoin address[5].

Given that the network needs to have knowledge of the full blockchain, this would either result in the blockchain being unacceptably large for network, or since storage of a transaction would factor into it's cost, the transaction fees would greatly increase and become impractical to use.

The prevent this issue, the HPP Network will implement a message storage system with the message contents permanently stored off the blockchain. Transaction output can be broken up into the message contents and coin transfer and so stored separately, with only the coin transfer permanently being stored on the blockchain. This is explained in detail in the following section.

There are other scaling measures which could be investigated, such as pruning older transactions to reclaim space[6]. We could also implement `sharding` into the blockchain to reduce the space occupied on any node, which has been proposed in some blockchains[7].

4.1 Message storage system

For the core HPP Network to operate, only coin transfers need to be held in the ledger that is the blockchain. It is greatly inefficient for the blockchain to permanently store the messages with the amount of unnecessary data duplication. As such we propose a system parallel to the blockchain which can permanently store the messages.

The intention is that message contents should remain on the blockchain for the most recent 10,000 blocks, which will be about half a week. This is so there is no cost to retrieving a recent message. Blocks before this are considered `old`.

The core features of this system are that:

- The amount of data any particular node stores is decided by the node itself. There will be nodes ran by the HPP foundation storing the entire history of messages, but nodes are not required to store any message history beyond the most recent blocks. It could be possible for nodes to subdivide the entire history and operate in a way similar to a distributed database, such as Cassandra[8], this would allow for a controlled replication factor and flexible quorum method among nodes operating in this way.
- There is incentive to store data. Whilst more recent transactions are still held on the blockchain, there is a message retrieval process to retrieve older transactions will come at a cost to the message receiver. This operates in a competitive way with a proof of work required to mine the message retrieval.
- Message data remains unforgeable. Instead of storing messages on the blockchain, the messages are replaced by a SHA-256 hash of the actual message contents to act as a checksum. As part of the message retrieval process, this checksum is compared against the retrieved result to validate the transaction. The checksum algorithm includes the one-time public key to further prevent identifying the message contents.

4.1.1 Message storage process

The process for moving messages to the storage system differs based upon the amount of message data the node wants to store and is as follows:

1) For an `old` block containing messages, the node calculates the SHA-256 hash (M_{hash}) of each transaction's message body and headers ($M_{contents}$).
2) The node replaces the message contents $M_{contents}$ with a key `MessageChecksum` with the hash M_{hash} as the value.
3) The node calculates the size of $M_{contents}$ as M_{size} and adds a key `MessageSize` to indicate the size message contents M_{size}.
4) The node decides whether it wants to store this block's messages. If it does not want to, the process ends here.
5) For a node wanting to store the messages, the message should be held in a system in which the data can be accessed by a key-value pair - the message hash M_{hash} as the key and the message contents $M_{contents}$ as the value.

The implementation of the key-value storage system is down to the node since we're only defining the protocol of being able to retrieve the message.

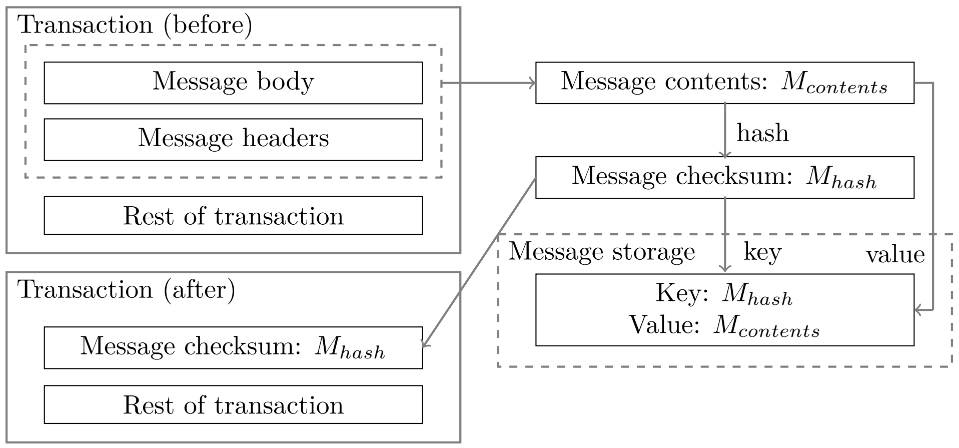


Figure 2: Message storage process

4.1.2 Message retrieval process

The process for a requesting user wanting to retrieve an `old` message whose contents has a hash of M_{hash} is as follows:

1) The requesting user broadcasts a message retrieval request signed by them to retrieve the message contents of a particular previous transaction. The message retrieval request should contain:

- The key they want to retrieve M_{hash}.
- The retrieval fee $F_{retrieve}$ to be received by the `message retrieving` miner who is retrieving the message contents and creates a `message retrieval` transaction.
- The transaction fee F_{trans} to be used by the `block creating` miner who is processing the `message retrieval` transaction into a block. This should be based upon the size of the message contents M_{size}.
- The signature ${Signature} to be used as the sender of the transaction.

2) A `message retrieving` miner detects this retrieval request and determines whether they've stored the message with key M_{hash}.
3) The miner retrieves the message contents $M_{contents}$ stored under the key M_{hash}.
4) The miner then has to mine a `message retrieval` transaction which has the following conditions for it to be considered valid:

- The original message retrieval request is included and is valid, it must have the signature of the original requesting user.
- The message contents $M_{contents}$ is included and the SHA-256 hash of this is M_{hash}.
- The miner has included paying themselves the retrieval fee $F_{retrieve}$.
- The miner has included a proof of work that takes a function of the other contents of the transaction.

5) The other nodes detect the transaction and verify the aforementioned conditions for the miner's transaction.
6) If the `message retrieving` miner's transaction is considered valid, the transaction

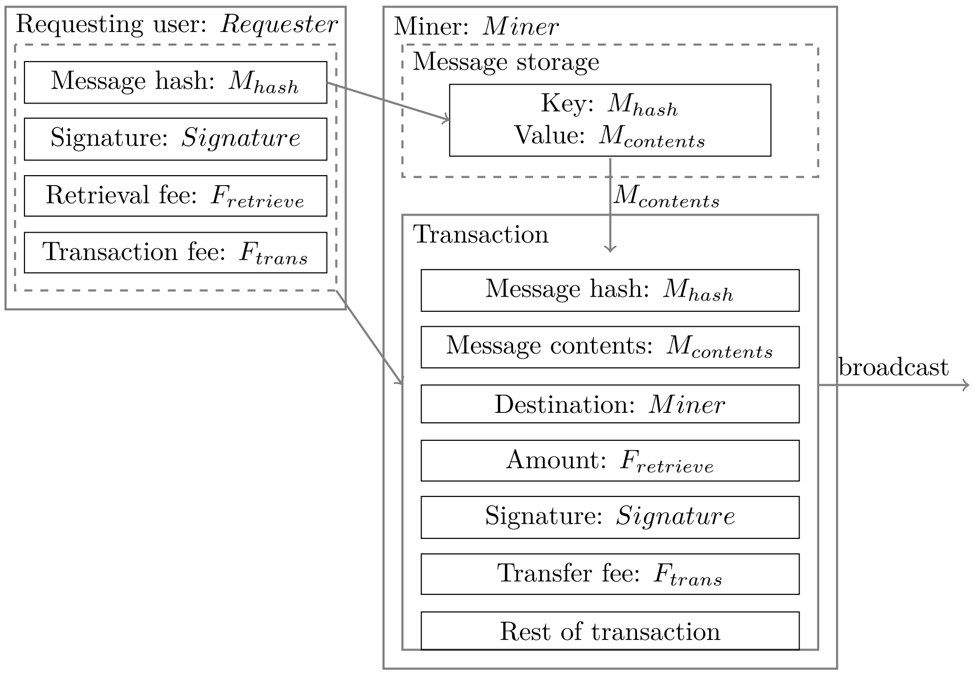


Figure 3: Message retrieval process

is added into a block and the `block creating` miner receives their transaction fee F_{trans}.

The message contents can then be retrieved by the requesting user for a limited amount of time. The proof of work difficulty should be tied to the main blockchain difficulty. The message size key is solely used by the message requesting user to estimate how large the `message retrieval` transaction will be, they are responsible for providing the fees for both mining the `message retrieval` transaction itself and the fee for adding this transaction to a block. Should the transaction fee be insufficient for the actual message size, the `message retrieving` miner could simply reject the message retrieval request.

The message retrieval transaction structure is discussed in more detail in the transaction section.

5 Message anonymity and authentication

The HPP protocol is built such that the outside observer will have no knowledge of who the receiver and the sender of a message is.

- For anonymously sending messages, a sender will use a one-time ring signature.
- To anonymously receive a message, unlinkable one-time public keys are used.
- The protocol also allows the sender to identify themselves to the receiver, as explained in the section on message headers.

So, together these guarantee complete anonymity without compromise.

## One-Time ring signatures

As explained, we propose the use of ring signatures to anonymously send messages.

Instead of the sender producing a signature which can be checked by just their own public key, the sender selects, at random, multiple public keys from the other users of the system. This provides a level of deniability to the sender's identity. By the sender including, say, one hundred other signatures then there are one hundred other users who could have sent the message[9].

Given we have a sender with a random secret x, the process is as follows:

1) The sender creates a public key P as $P = xG$ and the 'key image' as $I = xHp(P)$, where Hp is a hashing function.
2) The sender selects n other user's public keys as P_{set} and adds their own public key P to the set, where n is a standard across the network for the level of deniability.
3) For each key P' in the P'_{set}, the sender calculates two numbers: a and b

- For the sender's own public key, a is a random number and b is zero.
- For the other public keys, a and b are both random numbers.

6) The sender calculates L_{set} by mapping each key P' in the P'_{set} to $L = aG + bP'$ where G is the base point used to generate public keys from the private.

- For the sender's own public key, $L = aG$.

7) The sender calculates R_{set} by mapping key P' in the P'_{set} to $R = aHp(P') + bI$.

- For the sender's own public key, $R = aHp(P)$.

8) The sender generates the non-interactive challenge, a way of verifying the sender's legitimacy without revealing them that can be performed without them needing to interact with a verifier: $c = Hs(m, L_{set}, R_{set})$, where Hs is a cryptographic hashing function and m is a message.
9) The sender calculates (C_{set}) by mapping each key P' in the P'_{set}:

- For the other user's public keys, $C = b$.
- For the sender's own public key, $C = (c - \sum b) \mod l$, where l is the limit of the random number range.

10) The sender calculates r_{set} by mapping each key C in C_{set} and corresponding values a to r:

- For the other user's public keys, $r = a$.
- For the sender's own public key, $r = (a - Cx) \mod l$, where x is the random secret.

11) The sender calculates the signature $S = (I, C_{set}, r_{set})$.
12) The sender attaches S to the transaction as the `Signature` key.

For nodes on the network, the transaction must be verified. For the signature S, message m and public key set P'_{set}:

1) For each value P' in P'_{set}, C is C_{set} and r in r_{set}, the node calculates L'_{set} with $L' = rG + CP$

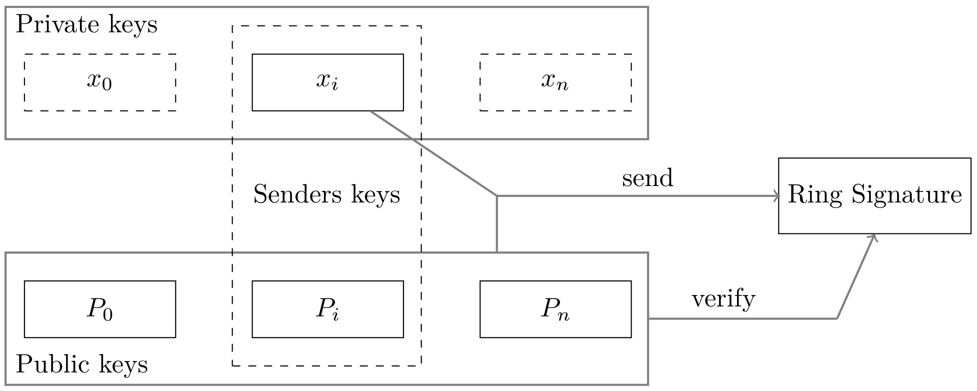


Figure 4: Ring signature algorithm

- For the public key of users other than the sender, this value $L' = aG + bP'$ and so $L' = L$
- For the sender's public key, secretly, the value $L' = rG + CxG = (r + Cx)G$

2) For the same set of values, the node calculates R'_{set} with $R' = rHp(P') + CI$

- For the other users' public key, this value $R' = aHp(P') + bI$ and so $R' = R$
- For the sender's public key, secretly, the value $R' = rHp(P) + CxHp(P) = (r + Cx)Hp(P)$

2) The node calculates the result of the non-interactive challenge, firstly by calculating $c' = Hs(m, L'_{set}, R'_{set})$
3) Then by verifying: $\sum C \mod l = c' \mod l$.
4) If the verification passes, the node further verifies that I has not been used before.

A full security analysis is left outside of this paper but we can see that L' and R' for the sender's public key have the same coefficient. If an attacker were to use random value instead, there would be no key matching this condition.

Since the image I can only be used once, this protects the HPP Network from double spending since x maps to one value of I.

5.0.1 Standardized ring size

To ensure privacy, the ring size needs to be standardized for every transaction sent. If senders were to use different ring signature sizes, it would in part be a fingerprint to their identities, especially since the habit would be for a sender to use the same size every time.

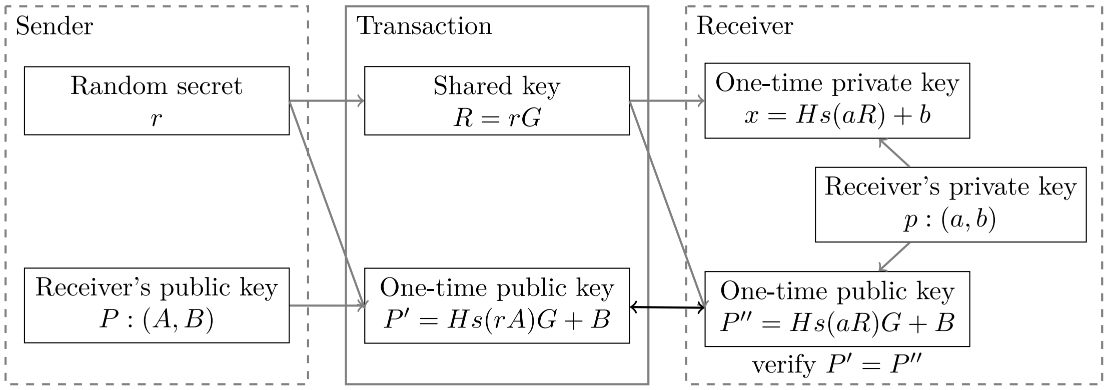
Say a particular sender were to always include twenty other randomly chosen public keys rather than one hundred, assuming this as the standard size. Other transactions with twenty other random public keys and this particular sender's key would very likely be coming from that sender, hence revealing their identity. If every user were to choose their own ring size then this would defeat the purpose of the ring signature entirely.

As such, HPP will have a standard ring signature size for transactions.

5.1 Unlinkable one-time public keys

Without one-time public keys, the receiver of a transaction on a blockchain is publicly known and so a complete transaction history of a particular user can be established. The only option to prevent this would be for the user to create multiple addresses and keep them unlinked.

To resolve this, the HPP Network implements one-time public keys, allowing a user to instead publish just a single address and anonymously receive unlinkable transactions by default[5]. The preconditions for this require that an HPP address be twice the size of a Bitcoin address, since the address constitutes two parts.



Sender
Random secret r
Receiver's public key $P : (A, B)$
Transaction
Shared key $R = rG$
One-time public key $P' = Hs(rA)G + B$
Receiver
One-time private key $x = Hs(aR) + b$
Receiver's private key $p : (a, b)$
One-time public key $P'' = Hs(aR)G + B$
verify $P' = P''$

Figure 5: One-time public key algorithm

The steps for the sender are as follows:

1) The sender takes the receiver's public key P and extracts out (A, B).
2) The sender generates a random secret r.
3) This is used to compute a one-time public key P' as $P' = Hs(rA)G + B$ where Hs is the cryptographic hash function Keccak and G is the base point used to generate public keys from the private.
4) The sender calculates a shared key R as $R = rG$.
5) P' is now set as the destination key of the transaction and R is set as the shared key.
6) The sender sends the transaction.

The steps for the receiver are as follows:

1) The receiver listens for transactions on the network and wants to check whether a particular transaction is theirs.
2) The receiver calculates aR from the shared key R in the transaction and from their private key (a, b).
3) This is used to compute the receiver's image key P'' as $P'' = Hs(aR)G + B$.
4) If P'' is equal to P', then this implies that rA, from P', is equal to aR, from P'', and so the transaction belongs to this receiver.
5) The receiver retrieves the one-time private key from $x = Hs(aR) + b$.
6) The one-time private key allows the receiver to spend the output and also unlocks the message in the transaction.

It should be noted that:

- Each destination is a public key that is generated from the recipient's address and random data. The random data allows the destination to always be unique and so it is possible to ensure that the outside observer doesn't know whether two transactions are going to the same destination.
- For a user to identify their transactions, they only use half of their private key (a, B) in the form of P'' above. This is known as the `tracking key` and it's uses are explained in the next section.
- In theory, a user could use a truncated address to allow any one on the network to identify transactions to be received by them. This is discouraged in preference to using a `tracking key` which allows for finer control.

5.1.1 Tracking key uses

If a user gives a third party their `tracking key`, the third party can detect which transactions belong to the user. This allows third party services to be set up to identify messages on the behalf of the user without allowing them to read the message. This approach is also useful when the receiver lacks bandwidth or computing power (smartphones etc.).

6 Transactions

We discuss the different transaction structures in this section.

6.1 Coin exchange transaction structure

The coin exchange transaction structure is as follows:

Key	Usage
TransactionType	Tells the network how to handle this transaction
Timestamp	Timestamp of the transaction
TimestampChecksum	Timestamp checksum to help the receiver
Destination	One-time public key as the receiver of the transaction
SharedKey	Random shared key for the one-time public key
Signature	One-time ring signature of the sender
TransactionFee	Transaction fee paid by the sender
Output	Output of the transaction

Coin exchange transaction structure

```
{
  "TransactionType": "coin_message_transfer",
  "Timestamp": "1502915687",
  "TimestampChecksum": "225:5000",
  "Destination": "b12bf48b48cc527b95",
  "SharedKey": "b8d66bea925ded44e98b",
  "Signature": "27b953b8d66be",
  "TransactionFee": "0.0001",
  "Output": {
    "Amount": 33
  }
}
```

6.2 Message transaction structure

The message transaction structure is simply an extension to the coin transaction structure and so the transaction type stays the same. Importantly, it adds message specific details under the `Message` key:

Key	Usage
Body	Content of the message
Headers	Additional information about the message
Attachments	Attachments to the message

The message body, headers and attachments are attached to the transaction and encrypted with the one-time public key. As such, the message contents is not visible to anyone except the receiver.

Message transaction structure

```
{
  "TransactionType": "coin_message_transfer",
  "Timestamp": "1502915687",
  "TimestampChecksum": "225:5000",
  "Destination": "b12bf48b48cc527b95",
  "SharedKey": "b8d66bea925ded44e98b",
  "Signature": "27b953b8d66be",
  "TransactionFee": "0.0101",
  "Output": {
    "Amount": 33,
    "Message": {
      "Body": "0bdc1df397615...",
      "Headers": {
        ...
      },
      "Attachments": {
        ...
      }
    }
  }
}
```

6.2.1 Message body

The message body represents the content of the message. The decrypted plain-text body will be prefixed with 4 zeroes. The intention of this is that a receiver of the message can check it was decrypted correctly.

### Message headers

The headers will contain additional information about the message. Some that will almost always set, but any other arbitrary data can be added here. For instance, a message header could contain a secret key as a way of whitelisting particular senders, to allow them to allows appear in a priority inbox. Additional headers can be applied by applications for the sake of driving innovation.

Commonly accepted headers:

Key	Usage
Sender	Optional Address of the sender
SenderSignature	Used with the Sender, to verify the Sender's address
CompressionMethod	Compression algorithm used on the message
ChainId	An ID for a message chain
ChainLinkId	Order of a message in the chain
WhitelistSecret	Secret shared to trusted senders

Example message headers structure

```
{
  "Sender": "2fd4e1c67a2d28fced849ee1bb76e7391b93eb12",
  "SenderSignature": "dabbdb94977298ff5a8d06a200ad7ec9",
  "CompressionMethod": "HPP_gzip_1.0.0.0",
  "ChainId": "b38d1343",
  "ChainLinkId": 1,
  "WhitelistSecret": "a83681de22db7ce16672f16f3"
}
```

6.2.1.1 Sender and Sender Signature header

By default, the protocol will stop a message receiver from identifying the sender, this is a potentially unwanted side effect of using the one-time ring signatures. To adjust for this, the protocol includes an optional method to identify sender to the receiver whilst still remaining anonymous to the rest of the network. The sender identifies themselves by the Sender header and proves it by the Sender Signature header.

Given we have a sender with a private key x and corresponding public key P:

1) Sender calculates H_{sig}, the SHA-256 hash of the transaction signature T_{sig}.
2) Sender calculates the sender signature S_{sender} encrypting H_{sig} with their private key x.

3) In the message headers, the sender attaches S_{sender} as the `Sender Signature` header and their public key P_{sender} as the `Sender` header.
4) Receiver retrieves their message.
5) Receiver calculates H_{sig}, the SHA-256 hash of the transaction signature T_{sig}.
6) Receiver verifies that the `Sender` header P_{sender} was used in the ring signature.
7) Receiver uses the `Sender` header P_{sender} to decrypt the `Sender Signature` S_{sender} to give H'_{sig}.
8) Checking the result H'_{sig} is the same as the `Sender Signature` header H_{sig} verifies the sender's identity.

Given the transaction signature T_{sig} changes with every transaction, as a consequence of the one-time public key feature, the sender's identity can not be forged. We leave it to the application layer to decide whether to show anonymously sent messages or not.

6.2.1.2 Compression method header

All message bodies and attachments will be compressed to reduce the amount of space required to send them. The message headers allow the sender to specify the compression used. This has been added to allow future variations of compression to be added to the network as innovation drives. Official HPP compression versions will have the format of: `HPP_NAME_VERSION`, with `NAME` taking on the algorithm name and `VERSION` loosely following semantic versioning.

6.2.1.3 Chain id and Chain link id header

A chain id can be shared between two addresses in a message transaction to represent an ongoing chain of communication, a conversation. The chain link id decides the order of the messages in that chain.

After the first communication with a particular sender, it may be preferential to send just the chain id and chain link id, instead of the sender address and signature. This would reduce the size of the transaction.

6.2.1.4 Whitelist secret

A user can use a whitelist secret to prioritize particular senders in their inbox. This could be used by the receiver to identify known senders, say within their social group. The process is as follows:

1) A user generates a secret and saves it on their application.
2) A user then gives this secret to a sender by some means.
3) A sender adds this secret into a future message transaction.
4) Applications should check for this secret in a message to determine a whitelisted sender.
5) Message is prioritized in the inbox.

This implementation allows for a user to have multiple whitelist secrets, with potentially different behaviours, and also decide to cancel a secret at will, say if a secret is given to whitelist a sender for a limited amount of time or if the user wants to retract the privilege.

6.2.2 Message attachments

The message attachments can be added to a message transaction. These should be be encrypted in the same way as the message body. They take the following structure:

Key	Usage
ContentType	MIME type of the attachment
CompressionMethod	Compression algorithm used on the attachment
Content	Contents of the attachment

Example message attachments structure

```
{
  "Attachments": [
    {
      "ContentType": "application/pdf",
      "CompressionMethod": "HPP_gzip_1.0.0.0",
      "Content": "c1f0fd905cc22888f5151dbfbedb48"
    },
    ...
  ]
}
```

#### Content type

This defines the content type for applications to understand how to open the file. These take the standard MIME types accepted in HTTP.

Common examples

- `application/pdf`
- `application/zip`
- `image/png`

6.3 Message retrieval transaction structure

The `message retrieval` transaction is used as part of the message retrieval process, and has the following structure:

Key	Usage
TransactionType	Tells the network how to handle this transaction
Timestamp	Timestamp of the message
TimestampChecksum	Timestamp checksum to help the receiver
Destination	The address of the miner who created this transaction
Signature	The signature of the requesting user
TransactionFee	Transaction fee paid by the requesting user
MessageHash	Hash key the requesting user wanted to retrieve

Key	Usage
MessageContents	Message contents matching the MessageHash
Output	Output of the transaction containing the miner's reward
DifficultyTarget	Proof of work target difficulty
Nonce	Used as part of the proof of work algorithm
TransactionHash	Hash of the rest of the transaction

Message retrieval transaction structure

```
{
  "TransactionType": "message_retrieval",
  "Timestamp": "1502915687",
  "TimestampChecksum": "225:5000",
  "Destination": "b12bf48b48cc527b95",
  "Signature": "27b953b8d66be",
  "TransactionFee": "0.0001",
  "MessageHash": "cfda6108187be6fe73b9d8f665c41cd2d61f1e798eeeb9270cea1e4efe648092",
  "MessageContents": "2007a5d5c28ba7626f104baa1db062b8de982ac281e3b305afb39acc8e04e647",
  "Output": {
    "Amount": 33
  },
  "DifficultyTarget": 2.000,
  "Nonce": "222bda451fd4",
  "TransactionHash": "c8794ab93c00de97d77fe2d27e49af9a0e0df9d1660cb2..."
}
```

For the validation conditions of this transaction:

- The miner has completed the message retrieval transaction by sending themselves the transaction fees with their chosen public key as the `destination`.
- They have provided a proof of work which is fulfilled by the `nonce` and `transaction hash`.
- They have included the original message retrieval request as the `signature` and `message hash`.
- They have included a `message contents` whose SHA-256 hash is the `message hash`.

7 Economics

7.1 Economics of a blockchain

Most blockchains work around the concept of a coin having intrinsic value to both the senders and receivers[10]. The blockchain ensures the growth of the coin's value by:

- Having competition to complete a proof of work function for the miners, and to a lesser extent, for a proof of stake function. This consumes fiscal resources of the miners, be it in electricity or hardware costs, to add intrinsic value to the network. Regulation, automatic or otherwise, for the release rate of the coins from blocks and

increasing difficulty of the proof of work function ensures a gradually increasing amount of resources are required to mine new coins.
- Having a transaction fee to transfer coins, meaning the sender is more reserved about spending their coins and so the coins become more scarce. Competitive transaction fees exist based upon the limited block space available, both immediately and in the overall running of the blockchain.
- Trust in the network in general. This tends to be driven in part by stability of the network, trust between the community and foundation behind the blockchain and in part by the continued incentive of stakeholders to drive new users, and hence demand, to the network.

In the general, there is incentive to increase demand and create scarcity[10].

## Economics of the HPP Network

For the HPP Network, we have to consider things differently. Coins are not the only thing that have an intrinsic value. Sending a message and getting someone's attention both have value for the sender.

- For a typical message, in the eyes of the sender, the value of the message is the same as it's transaction fee.
- For an attention bidding message, in the eyes of the sender, the value of the message being read by the recipient is the same as the bid. We assume that the bid will generally be much greater than the transaction fee.

There may also be value to the receiver of a message, beyond the receiving the bid, though that is hard to quantify. So for HPPCoin, there is the general value in holding the coin as with most blockchains but also the value of being able to use HPPCoin to bid for the attention of a message receiver.

However, for HPP to be used widely, messages without any bid should not be too expensive to send, so transaction fees need to be continuously low. This can be achieved by free market consensus of the network.

7.1.1 Message storage

We have to consider the value of storing the message. Since the blockchain is a ledger for not just coin transfers but also for messages, the space the messages occupy on network nodes, though temporarily, comes at a cost to the network nodes and so will also factor into the transaction fees as value to both the receiver and the sender.

The HPP Network has this considered the cost of space in it's design by ensuring that older transactions are put into storage to reduce the size of the overall blockchain.

With this in mind, long term message retrieval comes at a cost to the receiver and so, should be cheap. The message retrieval incentives should be regulated by the same adaptive proof of work difficulty as the block creation incentive.

7.2 HPPCoin

We require a coin to underlie the requirements set up by the HPP Network:

- Without some form of payment, mining is not incentivised and hence transaction processing can not operate. Messages will as such not be sent.
- Without bidding for a users message, the attention bidding functionality is not possible.
- Without transaction fees, spam messages will flood the network.

We propose HPPCoin, which has the following characteristics:

- The total number of HPPCoin is 1 billion (1,000,000,000), this creates scarcity in the amount of coins and in the block space, and so adding value to the coin.
- HPPCoin can be divided into units of 1 billion called HPPNano, this allows for transaction fees to be paid whilst keeping the value of the coin high.

7.2.1 Future of HPPCoin

We have taken into consideration the future of HPPCoin. In general, by imposing a limit on the total number of blockchain coins, the incentive for the miner in the distant future is no longer based upon the block reward and instead upon the transaction fees they would receive.

Given a fixed block size, one of two scenarios is possible:

- Transaction fees become excessive, since storage space on a block becomes highly competitive. In a message sending system, this is particularly a problem.
- Long delays in message delivery that prevent time sensitive information being delivered.

We avoid these scenarios by using an adaptive maximum block size (flexible soft-limit block size) and a proof work function whose difficulty varies as a function of the median block size, transaction fees and number of remaining coins.

8 Mining HPPCoin

We accept that for the network to be stable, mining has to be incentivised and that incentive has to be reliable. This requires that the value of the coin has to be either maintained or growing. It also needs for the block reward emission rate to be a smooth function and not suddenly changing. As with other blockchains, we allow transaction fees to be placed into any transaction.

The mining difficulty will automatically adapt to the network fees.

8.1 Transaction fees

In HPP, as with other blockchain systems, it comes down to the transaction sender to attach a transaction fee. The transaction fees tend to be influenced by two factors:

- The spatial size of the transaction. Given the soft-limited maximum size of a block, a larger transaction will take up more space in a block and so the sender would be expected to provide a larger transaction fee for the miner. Particularly important to HPP, since messages have greatly varying sizes.
- The speed the sender wants the transaction to be processed in. A smaller transaction fee will result in a larger delay in processing time since there is less incentive for a miner to include the transaction in a block.

It comes down to the sender to attach a suitable sized transaction fee in proportion to the size of their message and the speed they want the transaction to be processed in. The general going rate for a transaction, per byte, can be obtained from the network itself and so the sender can approximate how much they should be paying to get a message sent in a certain timeframe.

The market for blockchain space freely determines what senders are willing to pay to complete a transaction. The determining factor in HPP is different though. In most blockchains, the market is driven by how much senders are willing to pay for their money to be received quickly. On the other hand, HPP transactions can have no coin payment contained in them but still have a message, and this is likely to be the most common transaction scenario.

As such, the going rate for transaction fees represents how much senders value being able to send a message to each other, which will not increase much over time. So, as the value of HPPCoin increases, the relative transaction fees will decrease.

8.2 Block mining rate and reward emission

The HPP Network will aim for there to be about 120 blocks created every hour, or every 30 seconds, regulated automatically by the adaptive difficulty. This is to ensure message transactions are processed quickly.

The block rewards will follow a linearly decreasing function, with the aim that the block rewards will continue for about 15 years. We expect the block reward to decrease by one coin every week, or every 20,000 blocks.

8.3 Flexible soft-limit block size

Some blockchains allow for a flexible hard-limit to be placed on the maximum block size of the next block[11]. This is based upon taking the moving median of the last N blocks and doubling it, with N being a number agreed upon by the network. So if the median size for the last N blocks is M, the maximum size is $2M$. It averts the blockchain from bloating but still allows the limit to slowly grow with time if necessary.

With HPP, we do not place a hard limit on the size of transactions, since a sender may wish to send a message of any size and so, should be entitled to do so, so long as the attached fee is large enough. It is therefore possible to create an extremely large transaction, one that is magnitudes larger than other transactions and fill up a block just by itself.

To accommodate for this, HPP instead uses a flexible soft-limit, an adaptation on the flexible hard-limit:

1) For the last N blocks, we find the largest transaction in each block.
2) We then exclude those transactions from the overall block sizes.
3) We take the remaining block sizes and find the median M' of the remaining block sizes.
4) We then take a soft-limit of $2M'$.

With the soft-limit applied, the network allows a miner to exceed the block size limit by one part of a transaction. So, a miner can fit as many transactions into the block as they want without going over $2M'$ size limit.

With the remaining space, the miner can potentially add in one large transaction to take the block size of over the soft-limit. The transaction fee for all transactions including the overflowing transaction will be received by the miner.

This soft-limit incentives filling a block well and adding a one large transaction on the end of a block. This places an indirect restriction on the number of extremely large transactions that can occur on the network in any given timeframe, though these transactions will be rarer and also highly sought after by miners, so they should clear out of the network fairly quickly.

8.4 Proof of Work

### Proof of Work over Proof of Stake

To incentivise mining, we choose a proof of work function over a proof of stake function.

Proof of stake works by allowing only the holders of a limited subset of the overall released coins to mine the next block[12]. This is deterministically chosen and proven to be random, and hence fair. It also makes it a lot harder for any particular individual or group of individuals to obtain control of the network, since it requires them to hold more than 50% of the network's coins. Importantly, it works best whenever the majority of coin holders are online continuously.

However, we foresee that the majority of HPP Network coin holders will not be online nodes, which requires high bandwidth and computing power. Instead, we expect most coin holders will use applications that will occasionally connect to the network and download their messages. This will be facilitated by third party services, continuously online to help users identify their transactions by their `tracking key`.

A proof of stake function would give an unfair advantage to any stakeholders that are continuously online, and many of the coin holders will not be in a position to mine coins even when they are online, if they use low bandwidth and low power devices like smart phones.

At the blockchain inception, proof of stake is most vulnerable to particular individuals gaining control of the network. So it is less suited to being used at this point.

So for the stability of the network and for fairness, we decide to employ a proof of work function in the HPP Network.

8.4.1 Proof of work algorithm

We choose a solution and verification model for proof of work, based upon the following requirements:

- ASIC-resistance
- CPU friendly but not necessarily GPU friendly
- Fast verification to avoid a Denial of Service attack

HPP follows the principal that mining should be accessible to the public, and so should be possible to do so effectively on commonly found hardware. Application specific integrated circuits (ASIC) mining devices have risen in popularity because of their ability to solve certain proof of work functions orders of magnitude better than other hardware[13]. We choose a proof of work function that is not favourable to using these devices.

Though good GPUs are common place and GPU friendly algorithms exist, the algorithms specifically targeting GPUs typically suffer from an growing in-memory data set[14]. After a certain time, the dataset exceeds the memory capacity of GPUs and so forces miners on lower end GPUs to drop out. Given GPUs tend to have discrete tiers of memory capacity, say 2Gb, 3Gb and so on, this also has the side-effect that the hashing power of the network suddenly drops whenever the dataset size exceeds the next tier.

We believe that a CPU friendly approach is best for reasons of greatest accessibility. Whilst, bot nets are an issue to CPU friendly approaches, we consider this to be an unsolved problem. If the dilemma comes down to choosing between allowing ASIC devices or allowing bot nets, the latter still allows for greater penetration of mining to the public. In any case, a miner could invest a large amount into mining if they choose to, for instance, build a cloud based system; the difference is the scale of the trade off.

For HPP, we have decided to use an implementation of CryptoNight[15], believing it to be the strongest candidate that matches our criteria with other options considered[16][17].

### Adaptive difficulty

Our adaptive difficulty for the proof of work function changes the difficulty for every block, building on previous work on this area[18][19], and is implemented with the following considerations:

- Maintaining a consistent block rate over time, this is more important for the network's running in general.
- Ensuring that once transaction fees become the dominate source of income for miners, the average return doesn't drop off.
- To ensure the block rate doesn't suddenly drop over a block reward boundary. This requirement is less significant early on since the fractional change is small.

The HPP adaptive difficulty algorithm is as follows:

1) Take all the blocks produced in the last day, about 2900 blocks, as B_{set}.

2) For each block B in B_{set}, calculate the time spent to work done ratio r as $r = T_{spent}/W$ to give r_{set}.
3) Sort the time spent to work done ratios r_{set}.
4) Remove outliers by taking only the central 80% range.
5) Calculate the median time spent to work done ratio as r_{median}.

We cut off outliers from both ends, firstly, to account for the variance of the time spent and inevitable anomalies, secondly, to prevent maliciously distorted timestamps that aim to affect the proof work difficulty.

If the next block is to be on the same block reward as all blocks in B_{set}:

6) Use the time spent to work done ratio to determine the work done required to bring the median time spent to the target value T_{target} from $W_{required} = T_{target}/r_{median}$
7) Calculate the new proof of work difficulty required to meet the work done, $W_{required}$.

If the next block crosses the block reward boundary or any of the blocks in B_{set} crossed the boundary:

6) Calculate the median transaction fees per block as F_{median}.
7) For each block B in B_{set}, determine the expected payout ratio P to give P_{set} with $P = (R_{new} + F_{median})/(R + F_{median})$, where R is the block reward for the block and R_{new} is the block reward of the new block.

- This can be simplified, since the block reward is only changed once a week and the difficulty algorithm aggregates blocks over one day, there will be a maximum of two payout ratios - before and after the boundary.

8) Calculate the mean payout ratio $P_{mean} = \sum_{x \in P_{set}} x/|P_{set}|$
9) Use the time spent to work done ratio to determine the work done required to bring the median time spent to the target value T_{target} from $W_{required} = T_{target}/r_{median}$
10) Adjust the work done required to factor in the payout changes, $W'_{required} = W_{required}P_{mean}$
11) Calculate the new proof of work difficulty required to meet the work done, $W'_{required}$.

We use the payout ratio to consider the work that miners would accept doing for a decreased payout over the block reward boundary. Considering this will ensure a smooth transition at the reward boundaries.

9 Other HPP features

9.1 Mixed payment and messages

Given a message transaction is just an extension of a coin transfer transaction, we are able to transfer money and a message at the same time. The HPPCoin sent in a transaction is also hidden from the outside observers. This allows for a wide range of different uses such as:

- Something typical, like sending a payment to an online shop, with reference information for the shop stored in the message.
- Sending anonymous donations to a charity with a secret message.

- Attention bidding for a receivers attention, as explained below.

As such, innovation is driven by how users want to use the HPP Network. The core purpose will be to provide a person to person communications platform but the network will enable other uses.

9.2 Attention bidding

There is economic value to getting someone's attention, such as for companies via advertisements, or to course public opinion via media headlines.

With mixed payment and messages in mind, HPP allows for message senders to bid for a receivers attention. By providing larger coin transfer, applications will sort the messages by order of the highest bidder. This way, if you want your message to be seen by an important individual, say a celebrity or politician, you attach a higher bid.

The effect of this is two fold:

- Popular public figures can be paid for their attention.
- Individuals who would otherwise be isolated from a celebrity, entrepreneur or politician they'd like to connect to, are able to.

## Spam reduction

Most spam campaigns will be uneconomical for the senders on the HPP Network since:

- Message transaction fees increase with the length of the message, making sending large messages multiple times expensive.
- Attention bidding means that would mean that even after sending a spam message, it will need to have a high enough bid to be read by the receivers.

As such campaigns with little to no value to them, scams, widespread offers on general products for instance, will not be economical. This would leave targeted campaigns that are narrowly directed at particular individuals or audiences.

9.3 Efficient message identification

Since the transaction receivers are anonymous in the HPP Network, to be able to detect whether a particular message transaction belongs to a user, the usual implementation would require that the user check every single transaction on the network and then verify whether it belongs to them. This is an expensive process.

In other communications systems based upon the blockchain, when a message is sent, a source node will broadcast messages to all nodes and will keep broadcasting messages until a confirmation from a reader is sent. This is done in a exponentially increasing time frame to ensure that the network is not flooded[1].

HPP does not implement this method since:

- If a reader is offline for a long time, it may take a couple of days before someone receives their messages given the exponential time frame.

- It requires the sender to keep broadcasting the message, so requiring the sender to stay online even after the message has formed part of a block.
- In confirming the broadcast message as having been accepted by a receiver, it in-part identifies the receiver.

The HPP Network does not take the a confirmation of reading approach to solve the message identification issue. Instead, there are a couple of ways in which HPP is built to make message identification more efficient:

- By allowing a third party to know their `tracking key`, a user allows an always online service to detect messages for them. So, messages can be immediately given to the user when the transactions are added to the block. This deflects the responsibility.
- By the timestamp checksum included in the message transaction, the user filter out the transactions before having to perform a more expensive verification on the remaining transactions. This is preferred since it can be used in the absence of a third party.

9.3.1 Timestamp checksum

The timestamp checksum is attached to a transaction so users are given a computationally cheap way to filter all transactions and pick out the ones that could potentially be with them as the intended recipient.

It is used by the transaction sender as follows:

1) Given the timestamp of the transaction T, the receiver truncates this to the nearest hour by $T_{hour} = T \mod 3600$.
2) Given the hourly timestamp of the transaction T_{hour} and the public key of the receiver P, the sender calculates the signature of the timestamp T_{sig}.
3) The network provides an updated value for the timestamp modulus T_{mod}.
4) The sender further calculates the checksum C as $C = \sum T_{sig} \mod T_{mod}$.
5) The checksum is attached to the transaction as the timestamp checksum in the following format $C : T_{mod}$.

The checksum is used by the receiver as follows:

1) The receiver picks up a transaction, uses the hourly timestamp of the transaction T_{hour} and their public key P' to calculate the signature of the timestamp T'_{sig}.
2) With the timestamp modulus provided in the timestamp checksum of the transaction, taken as T_{mod} from the timestamp checksum $C : T_{mod}$, the receiver calculates their checksum C' as $C' = \sum T'_{sig} mod T_{mod}$.
3) The receiver compares their checksum C' to C. If they are the same, they are a potential recipient. If not, the transaction isn't theirs and the process stops here.
4) The receiver then verifies whether the transaction is actually theirs.

With these processes in mind:

- Given N users on the network, a timestamp checksum created with modulus T_{mod} will have on average N/T_{mod} potential receivers for the transaction, so the actual receiver could be one of N/T_{mod} addresses.

- Given the random nature of the signature, the timestamp checksum will have a uniform distribution. In other words, the number of users for a particular checksum should be the same for any checksum.
- The checksum will not change for the following hour, so a user only needs to calculate the checksum once for all transactions within an hour instead of once for every transaction.

This concludes a way in which, with the timestamp checksum, the user can reduce the transactions they have to verify as theirs by a factor of the timestamp modulus and still maintaining a high degree of anonymity. Since the timestamp modulus is controlled by the network, it can be increased as the number of users on the network increases, so the balance of anonymity and required transaction verifications is also maintained.

## Real-life user verification

HPP allows a user to attach a real world identity to their address. The network protocol implements different adapters to verify a user by external and publicly confirmable means. Nodes receiving a particular verification broadcast can use a network adapter to verify the claim and so conclude the relationship. With this in mind, we will demonstrate how an adapter for Twitter can be used to securely attach a user's Twitter account to their HPP address. The process is as follows:

- Anonymous user creates their public P and private p keys.
- Using the user's Twitter account T and their private key p, the user calculates the signature of their twitter address $S_{twitter}$.
- User tweets from account T a publicly accessible message in an agreed format, such as: "Hey I'm on #HPPNetwork, address:P proof:$S_{twitter}$". All contents of the tweet other than the address and proof can be ignored. The format of this is flexible and should be varied for different locales.
- Nodes will automatically pick up the tweet and make a `claim` that the twitter user and address are related, or the user themselves can broadcast the `claim`. This includes a link to the public tweet `proof`.
- Nodes verify the `claim` and `proof` by checking for the public tweet themselves. Applying the public key P to reverse the signature $S_{twitter}$, will result in the twitter account name T.
- We leave it up to the application layer to then alias twitter account names as HPP user networks addresses.

This concludes an unforgeable way of allowing a user to verify their real life identity via Twitter. Similar adapters could be built for other publicly accessible systems.

9.3.2 Masking popular users

A consequence of combining real-life user verification and ring signatures is that any sender can include a public figure's address as one of the keys in the ring signature. Users could be motivated to do this to mask another users transactions among their own, say a whistleblower. If the whistleblower is masked by enough other senders, then their actual messages will go through with strong plausible deniability.

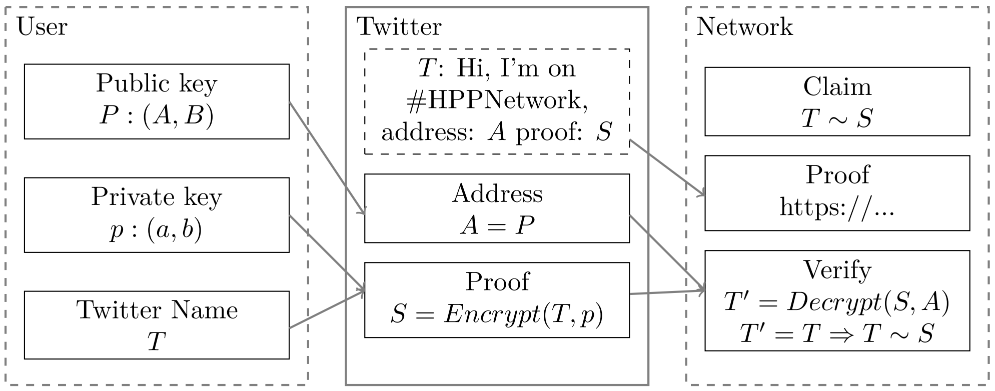


Figure 6: Twitter verification algorithm

10 Conclusion

We have presented how the HPP Network can create a new communications system. Instead of having to use obscure hashes as a form of identity, we can use verifiable human identities, whilst still allowing for complete privacy. We have shown how innovation can thrive from use of this network and a presented a range of convenience features that will come as standard.

References

[1] Jonathan Warren. Bitmessage - A Peer-to-Peer Message Authentication and Delivery System *https://bitmessage.org/bitmessage.pdf*

[2] Nicholas van Saberhagen. CryptoNote v 2.0, *https://cryptonote.org/whitepaper.pdf*

[3] Andreas M. Antonopoulos. In *Mastering Bitcoin*, Chapter 7 - The Blockchain.

[4] Blockchain Luxembourg S.A.R.L. Blockchain Size *https://blockchain.info/charts/blocks-size*

[5] Nicholas van Saberhagen. In CryptoNote v 2.0, *https://cryptonote.org/whitepaper.pdf*, Section 4.3 - Unlinkable payments

[6] Satoshi Nakamoto. In Bitcoin:A Peer-to-Peer Electronic Cash System, *https://bitcoin.org/bitcoin.pdf*, Section 7 - Reclaiming Disk Space

[7] What does Monero's scaling road map look like? *https://monero.stackexchange.com/questions/2885/how-would-sharding-work-in-monero-to-help-with-scaling*

[8] Avinash Lakshman, Prashant Malik. Cassandra - A Decentralized Structured Storage System *http://pages.cs.wisc.edu/ akella/CS838/F12/838-CloudPapers/Cassandra.pdf*

[9] Nicholas van Saberhagen. In CryptoNote v 2.0, *https://cryptonote.org/whitepaper.pdf*, Section 4.5 - Standard CryptoNote transaction

[10] Dylan Bargar. The Economics of the Blockchain: A study of its engineering and transaction services marketplace (2016). All Theses. Paper 2417.

[11] Nicholas van Saberhagen. In CryptoNote v 2.0 *https://cryptonote.org/whitepaper.pdf*, Section 6.2.2 - Size limits

[12] Ethereum Project. Proof of Stake FAQ *https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ*

[13] Torbjørn Langland, Kristian Klomsten Skordal. Mining Bitcoins using a Heterogeneous Computer Architecture, NTNU Trondheim. Section 2.3 - Evolution of Bitcoin Mining Hardware.

[14] Ethereum Stack Exchange, Ethereum DAG size predictions *https://ethereum.stackexchange.com/questions/426/what-is-the-current-dag-size-when-do-we-expect-to-hit-gpu-limits*

[15] BitCoinWiki. CryptoNight *https://en.bitcoin.it/wiki/CryptoNight*

[16] Alex Biryukov, Dmitry Khovratovich, Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem, Ledger (v2. 2017)

[17] CPU Coin List. Cryptocurrency Algorithms *http://cpucoinlist.com/cryptocurrency-algorithms*

[18] BitCoinWiki. Adaptive Difficulty *https://en.bitcoin.it/wiki/Adaptive_difficulty*

[19] Nicholas van Saberhagen. In CryptoNote v 2.0 *https://cryptonote.org/whitepaper.pdf*, Section 6.2.1 - Difficulty