

---

# SWEET-GOSSIP - P2P PROTOCOL FOR THE *GIG* ECONOMY

---

A PREPRINT

**Sonof Satoshi\***  
Public Key (GPG) 7  
Dont Trust, Verify  
[www.donttrustverify.org](http://www.donttrustverify.org)  
[sonof.satoshi@donttrustverify.org](mailto:sonof.satoshi@donttrustverify.org)

January 19, 2023

## ABSTRACT

Sweet-Gossip protocol is a P2P, mobile-first, Proof of Work protected, gossip protocol built on top of the Lightning Network that enables message broadcast (job proposal) and replies (job offer). It uses a game-theoretic approach to preserve its properties that are aligned with the Bitcoin ecosystem.

**Keywords** GIG Economy · Gossip Protocol · Distributed System · P2P Network · Bitcoin · Lightning Network

## 1 Motivation

GIG economy refers to the work done by casual workers coordinated by a software system. At the time of this writing, the end-customer of the Gig economy directly interacts with the centralized, cloud-based platform (app). This platform is also used to pay for the services after the job is done to the platform, which in turn is sharing the revenue with the assigned GIG worker. The actual job is done by the GIG worker for their customer, making the online platform a tool that supports and manages the effectiveness of the job.

This kind of cybernetic system uses the human power of GIG workers managed by AI, to extract value for the company shareholders. The AI component of the platform uses behavioural data represented as all the user (both GIG workers and customers) interactions with the platform to maximize the total revenue generated by the system and the underlying company that operates it. To optimise the global goal, which is the company revenue, the platform is implementing gamification and for-purpose misinformation techniques. It is possible because GIG workers and customers have no other option but to trust the platform for its efficiency and the underlying operation of the platform is opaque to users and configurable only by the operating central company. Therefore the platform operator:

1. dictates the revenue sharing and can change this anytime dependently on the socioeconomic circumstances without giving any reason to GIG workers
2. punish workers that are not behaving properly, that is aligned with company benefit, e.g. by blocking them from access to the platform

We are proposing a P2P protocol designed for the Gig economy that, by eliminating the need for central online platforms, will create a new decentralized, P2P GIG economy. Gig workers will be engaged directly by the end customer and can accomplish their tasks and earn money on a free market without the need for the existence of a central organization or any other trusted third party.

Lack of central organization also means that a minimal volume of data is shared between GIG workers and end-customers, just enough to fulfil the job according to the protocol-driven off-chain smart contract that uses P2P money i.e. Lightning Network [6] on Bitcoin [5], therefore forming layer 3 protocol from the Bitcoin perspective.

---

\*We are the children of Nakamoto

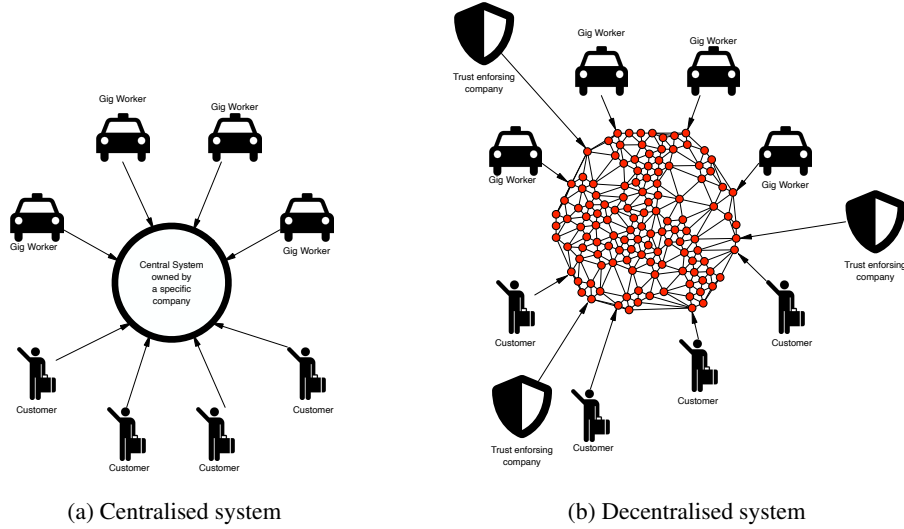


Figure 1: Centralised vs Decentralised Gig-economy

## 2 Sweet-Gossip P2P Network

Sweet-Gossip P2P Network is a global, symmetric, P2P network, meaning that there is no direct need to run any operation critical services in the cloud or any other centralised computing environment. Sweet-Gossip node is a software module that is run by every device that uses Sweet-Gossip protocol and forms a basis of communication. Sweet-Gossip nodes can be implemented as apps and run efficiently on cheap modern mobile devices. The need for implementation of supporting services that are cloud-based or edge-computing-based helps make the service more user-friendly but is never critical for the network operation.

It is important to state explicitly that we are not inventing any new coin or crypto token, but rather we are speaking about how Sweet-Gossip protocol forms a layer 3 protocol on top of the Lightning Network (being itself a layer 2 network sitting on top of Bitcoin network), therefore if any, the Bitcoin is a native token of the Sweet-Gossip Network.

Sweet-Gossip P2P Network preserves:

- P2P Symmetry - every node does the same thing
- Permissionlessness - anyone with internet access can join Sweet-Gossip P2P network
- Mobile first - the cost of running Sweet-Gossip node is marginal on modern mobile devices. Also, the protocol handles mobile connectivity issues.
- Privacy - the communication is encrypted
- Anonymity - any personal information about the people behind the nodes is hidden
- DDos and Spam protection - it uses Proof of Work (PoW) and/or micropayments to protect the network from DDos and Spam
- Sustainability - the protocol is designed so all its participants benefit from joining the network
- Implicit punishment - the protocol do not explicitly punish dishonest participants, but rather makes honest participant benefit more than dishonest ones

Sweet-Gossip Protocol is a gossip protocol [7], that allows a network to broadcast in a similar way to gossip spreads. Assuming that each sweet-gossip node is connected to its peers and that the network graph is connected, each node works independently and in the event of receiving a message that needs to be broadcasted. It selects several peers and sends the message, in its owner's interest, to peers making the message spread over the network like gossip (Figure 2). There inevitably occurs a situation that, if some node will not send the message to all of its peers, some nodes will not receive a broadcasted message even if a network graph is connected (e.g. node C on Fig 1.), but from the game theoretic perspective it will not be a beneficial situation, so this is up to the network operators to make the flow as efficient as possible.

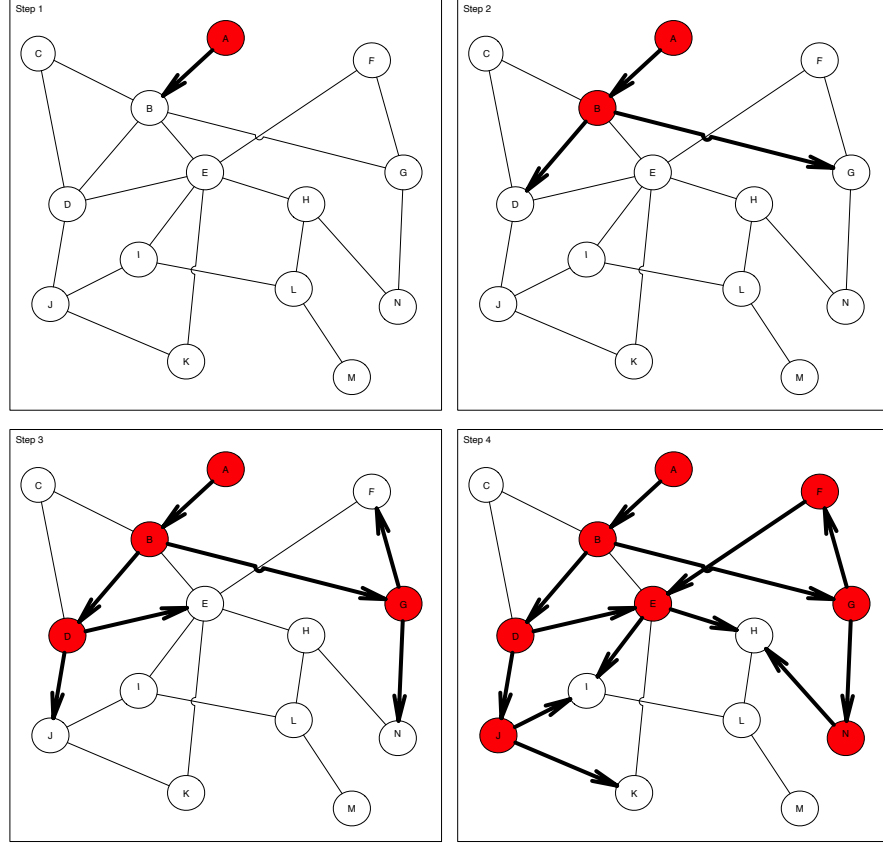


Figure 2: The intuition behind gossip protocol

Sweet-Gossip is a protocol, meaning that it only specifies the minimal set of rules to make it beneficial for all the nodes. It doesn't say explicitly how the network node should be implemented. The node implementation is free to do whatever is best to make it beneficial for the node owner.

### 3 The protocol

Sweet-Gossip protocol has a single purpose: to broadcast a job proposal (topic) to interested parties and collect job offers (reply messages) from interested contractors. Economically, the customer is interested in exchanging their money for the service, while gig-worker is interested in being paid for the job done. The network to sustain its existence needs to reward broadcasters for quality of the broadcasting. Therefore the protocol is constructed in a way that the customer broadcasts a topic, that contains anonymous job description. This job description is delivered by the network to the gig-worker that respond with network encrypted reply message. The message is delivered back to the customer that can verify basic properties of the gig-worker with their digital certificate but to decrypt contact information of the gig-worker is obligated to pay the network.

For sake of clarity we use the following naming convention:

1. Topic - the job proposal broadcasted through the network
2. Reply Message - the job offer for specific topic sent from the contractor
3. Peer - any gossip network node. Every peer maintains a list of their peers.
4. Sender - Peer that is the source of the topic. From the Gig economy point of view, it is a customer.
5. Originator - Peer that is currently broadcasting the topic to its peers.
6. Middleman - Peer that is passing the broadcasted topic further as well as bringing back the reply message

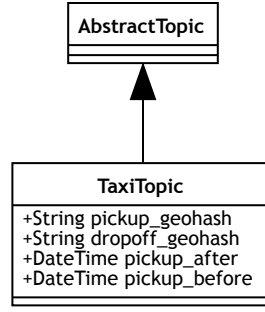


Figure 3: The topic frame

7. **Replier** - Peer that is replying to the broadcasted topic with reply message.

We assume that nodes of the sweet-gossip network are already connected to their peers via some internet transport protocol (e.g. TCP, UDP with or without hole punching [4], mobile mesh etc.) and the other peer is also accepting sweet-gossip protocol. How the nodes discover their peers is not a part of the protocol.

In short the sweet-gossip protocol can be summarised as follows:

1. **Asking for broadcast** If the originator (e.g. Node A) wants to broadcast the topic first step is to ask its selected peer (e.g. Node B) how about condition of its cooperation. If the middleman accepts this kind of topics, it replies to the originator with specific POW properties the originator needs to provide to be able to broadcast the topic with using this specific middleman.
2. **Broadcast with POW:** In order to use this middleman the originator must compute hash (e.g. SHA256) that is less or equal to the specific target for the specific POW scheme. The computed POW is passed with the topic to the broadcaster and if the broadcaster validates the hash it broadcast the topic to its peers.
3. **Replying:** If the middleman instead is interested in accepting the job it becomes the replier. Replier constructs the Reply Message. The Reply Message contains all the information that is required to pay for the reply message delivery to all the middlemans involved. The reply message is passed back to the originator
4. **Paying for the reply message:** Once the message reaches the originator, the originator needs to pay the network using the specific payment method. Once paid the reply message is revilled and the originator is able to begin direct communication with replier.

### 3.1 Topic

The topic is a data-structure that defines the basic requirements for the job. It is application specific and by design it should not reveal any information about the originator allowing for their identification.

Lets use a Taxi-app as an example Figure 3. The topic of this taxi-app has a form of geohash and time interval describing from where and when the ride can be executed. Geohash here is a way of encoding a specific geographical place (geographical cell that has a form of rectangle) in a form of a string where the lenght of the geohash determines its precision.

For example Legal Services Council in Sydney is located at the following coordinates latitude= -33.8647 and longitude=151.2096 and the corresponding geohash of precision 7 is equal to 'r3gx2g5'.

The precision of geohash determines the size of the cell (Table 1) and to be useful for the Taxi-app it needs to be atleast 7 so the cell has size lower than 200m (see table below)

On the other hand we dont want to be too specific and we might want to restrict the size of geohash to at most 8, so it is not possible to precisely locate the originator (customer) at this stage, but on the other side the precision is enough for the taxi driver to accept/reject to job.

Geohash length	Cell width	Cell height
1	$\leq 5,000\text{km}$	$\times 5,000\text{km}$
2	$\leq 1,250\text{km}$	$\times 625\text{km}$
3	$\leq 156\text{km}$	$\times 156\text{km}$
4	$\leq 39.1\text{km}$	$\times 19.5\text{km}$
5	$\leq 4.89\text{km}$	$\times 4.89\text{km}$
6	$\leq 1.22\text{km}$	$\times 0.61\text{km}$
7	$\leq 153\text{m}$	$\times 153\text{m}$
8	$\leq 38.2\text{m}$	$\times 19.1\text{m}$
9	$\leq 4.77\text{m}$	$\times 4.77\text{m}$
10	$\leq 1.19\text{m}$	$\times 0.596\text{m}$
11	$\leq 149\text{mm}$	$\times 149\text{mm}$
12	$\leq 37.2\text{mm}$	$\times 18.6\text{mm}$

Table 1: Geohash precision

Certificate
+Bytes public_key +String certificate_name +Object certificate_value +DateTime not_valid_after +DateTime not_valid_before +String cert_auth_name +Bytes cert_auth_signature

Figure 4: The digital certificate

### 3.2 Digital Certificates

Every gig economy environment needs to be safe for both customer and gig-worker. Safety means here the ability to have a level of trust that the other party will not violate civil rights of the other part during the service delivery either if it is a giving a ride, delivering food or programming website. The way to implement a physical levels of trust in the internet is done using Digital Certificates implemented as public-key certificate (e.g. X.509 certificates). These certificates are issued by certification authorities that can be either trusted 3rd parties, or communities. For a taxi-driver the minimal certification requires having valid driving licence and no criminal record. The trusted 3rd party can issue this kind of certificate and by signing it with its private key so anyone can verify that the specific certificate was truly issued by this trusted 3rd party. If the certificate is revoked the information about it is published by the trusted 3rd party in form of a revocation list. Public key certificates contain also a public key of the certified person, so it is possible use it to encrypt a message that is targeted for this person and verify their signatures.

On the other hand we don't want to be too specific and we might want to restrict the size of geohash to at most 8, so it is not possible to precisely locate the originator (customer) at this stage, but on the other side the precision is enough for the taxi driver to accept/reject to job.

### 3.3 Asking For Broadcast

The first step of sweet-gossip protocol is to send the AskForBroadcastFrame to the potential broadcaster.

AskForBroadcastFrame contains ask-identifier and digitally signed RequestPayload. signed RequestPayload is made of unique payload\_id, topic (e.g. TaxiTopic), sender certificate and sender signature obtained by signing the Request-Payload with the sender private key that is complementary to the public key stored within the sender certificate. Anyone can verify the Request Payload by validating its signature with sender public key from the certificate. Sender certificate can always be verified using public key of the certification authority and checking its published revocation list.

Ask-identifier allows for the frame identification during the originator-middleman ping-pong communication (see figure below). In the gossip protocol it is possible that the same broadcasting message hits the same gossip node many times, so payload\_id is to remain unique identifier that allows to determine this situation limiting these situations to minimum.

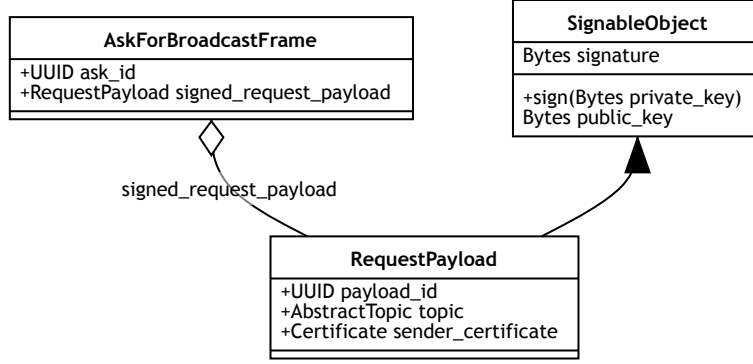


Figure 5: The Ask For Broadcast Frame

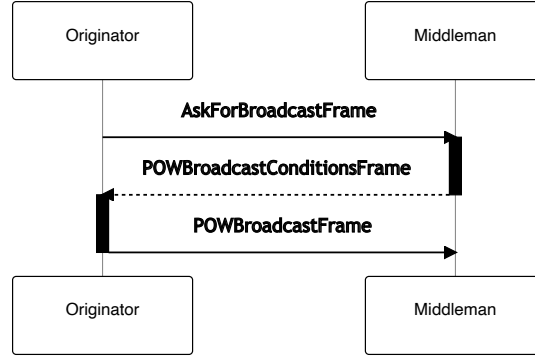


Figure 6: Ping Pong Sequence

It is the requested responsibility to ensure the uniqueness of `payload_id`, risking if it is not unique it will be lost during the broadcast as other nodes can decide that it was already broadcasted if the `payload_id` was already seen before.

### 3.4 Proof Of Work (POW)

Message broadcast is protected in sweet-gossip with the idea of Proof of work, introduced in HashCash [1] and famously implemented in bitcoin mining but originally introduced to limit the email spam. The thinking here is that if the originator needs to take some significant computational cost to be able to send the message it will significantly reduce the possibility of DDoS attacks. There are many possible POW schemas, here we are considering SHA256 hash based POW, similar to the one implemented in the bitcoin network. In short, given the topic the middleman decides how complex POW is required to be computed by the originator to allow him for further spreading of this topic. The task is to compute the hash of the `BroadcastPayload` so the hash itself is lower or equal to specific target. The larger target is the more complex the computation becomes. On the other hand, once the hash is computed, it is easy to verify that it fits into specific target, so the broadcaster has an easy task here to make sure that the originator has done the work to compute the correct hash.

### 3.5 Onion-routing

Sweet gossip is using onion-routing technique to hide the message reply route from the participating middlemen. During the broadcast phase the onion grows layer by layer. Active peer appends its address to the onion and is using public key of the next peer to encrypt the new onion, therefore only the next peer can decrypt that layer of the onion. Once encrypted the onion is passed to the next peer.

This way of constructing the onion allows then to peel the onion back to the sender through the network in a way that none of the nodes knows the sender nor the distant peers.

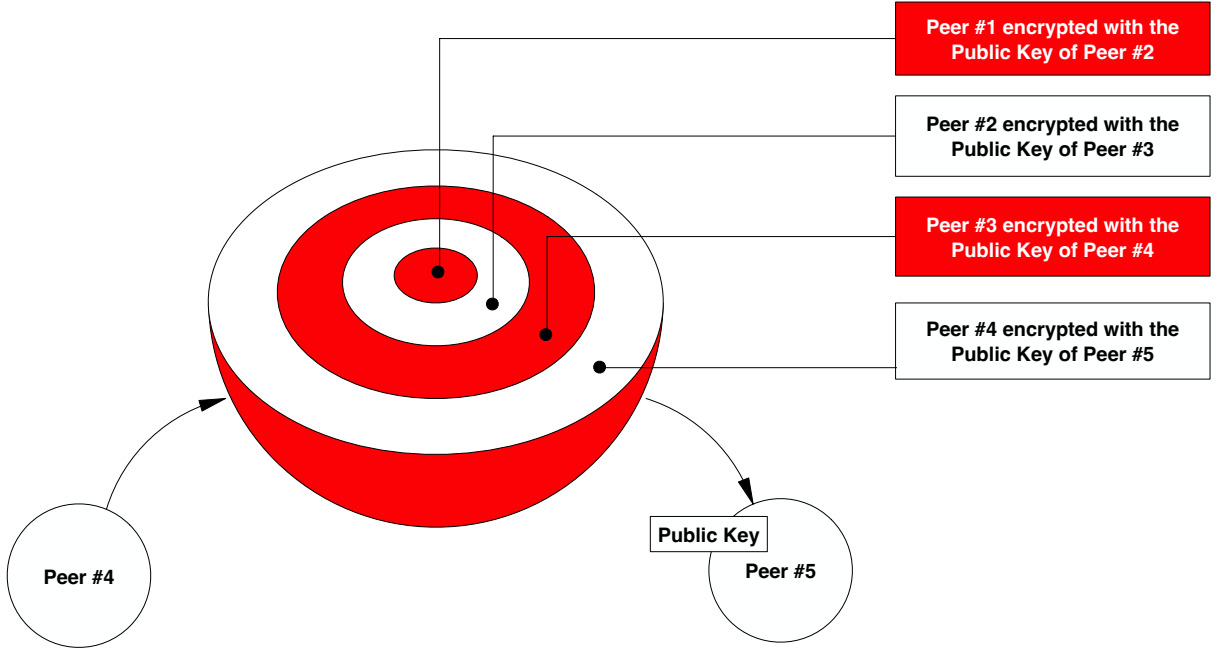


Figure 7: Onion-routing

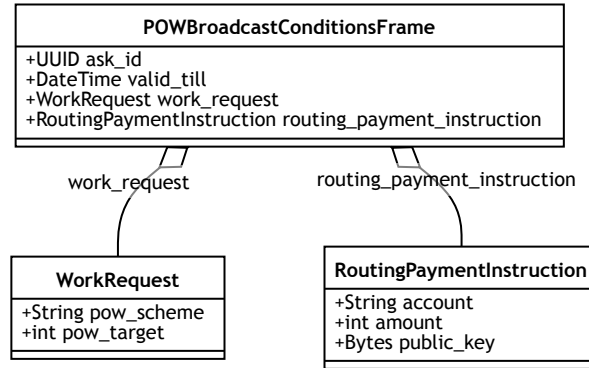


Figure 8: POWBroadcastConditionsFrame

### 3.6 Broadcast with POW

If the middleman accepts the topic specified in the AskForBroadcastFrame, it sends back the POWBroadcastConditionsFrame. This frame describes the properties of POW expected to be computed by the originator and payment instructions expected by the peer for delivering the reply.

Starting with ask\_id, that matches with AskForBroadcastFrame, and valid\_till timeout meaning that the middleman will wait only till the specific time for the POWBroadcastConditionFrame from the originator contains also WorkRequest that describes properties of POW. Routing payment instruction consists of the anonymous account and the amount to be paid. The amount here means only the price for this specific broadcast and the entire price of the reply is given as a sum given by all the middlemen that participate in this gossiping activity.

Originator is replying with POWBroadcastFrame that is also marked with corresponding ask\_id. The main part is a broadcast payload that contains original signed request payload (the one that was a part of AskForBroadcast and was already signed by the originator), onion-routing and routing payment instructions namely: backward\_onion and routing\_payment\_instruction\_list, both will be discussed here later.

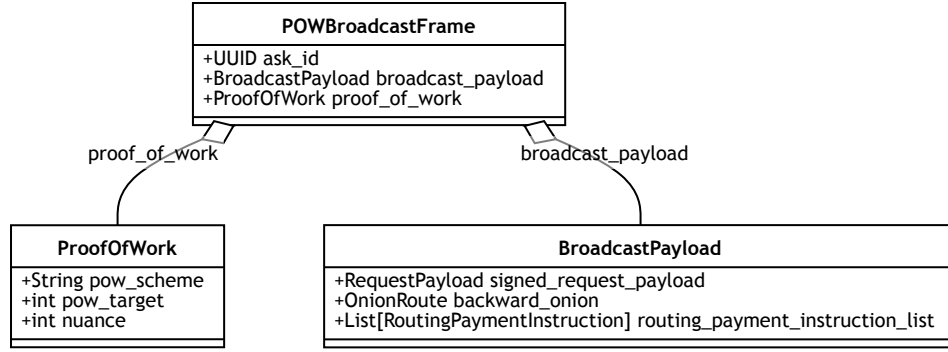


Figure 9: POWBroadcastFrame

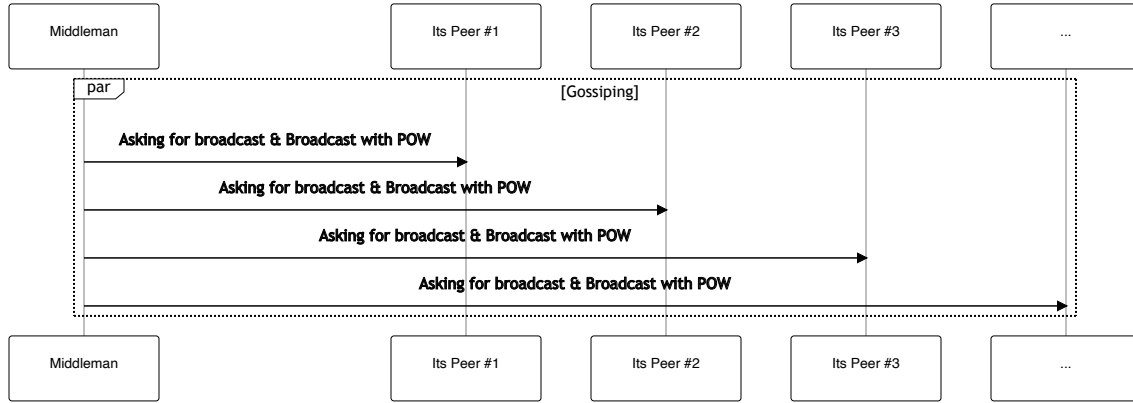


Figure 10: Broadcast Sequence

POWBroadcastFrame also contains ProofOfWork that contains a hash value (nuance) that fits below pow\_target for the specific pow\_scheme, and was computed as a hash of broadcast\_payload part, therefore middleman can easily verify nuance value by computing hash of broadcast\_payload and checking if it is lower or equal to the pow\_target.

Each step of the broadcast involves passing specific Broadcast Payload that consists of Request Payload that is never changed and protected by the cryptographic signature. Additionally, every Broadcast Payload has growing routing payment instruction list as well as adds new layer to the onion.

In the gossip protocol nodes are randomly selected from the list of all the known peers of the originator. This number is sometimes referred as fanout [7] of the gossip protocol. Once selected the broadcasting process is performed.

### 3.7 Lightning network, HODL invoices, payments, preimages and payment-hashes

Lightning network is a layer 2 network built on top of bitcoin network that allows for cheap and fast micropayments. It is built around concept of channels. Once channel is opened (that usually means funding it with some amount of BTC), it can be used to issue the invoice and pay the invoice. Payment generates a proof. This idea is extended with cryptographic concept of preimage for payment hash in the following way:

1. Invoice is issued by the issuer and as one of the fields it contains specific payment hash. Payment hash is a hash of preimage that itself is a number known only to its creator, that in case of HODL invoices is usually a different actor: a settler. The payment-hash is the only thing that is exposed on the invoice.
2. Payer is paying the HODL invoice under condition of having preimage published. In other words, the payment means that the invoice is paid if the preimage is revealed and that this preimage has the payment-hash that was presented on the invoice. Once paid the invoice is called Accepted.
3. Accepted invoice is Settled once the preimage is revealed.



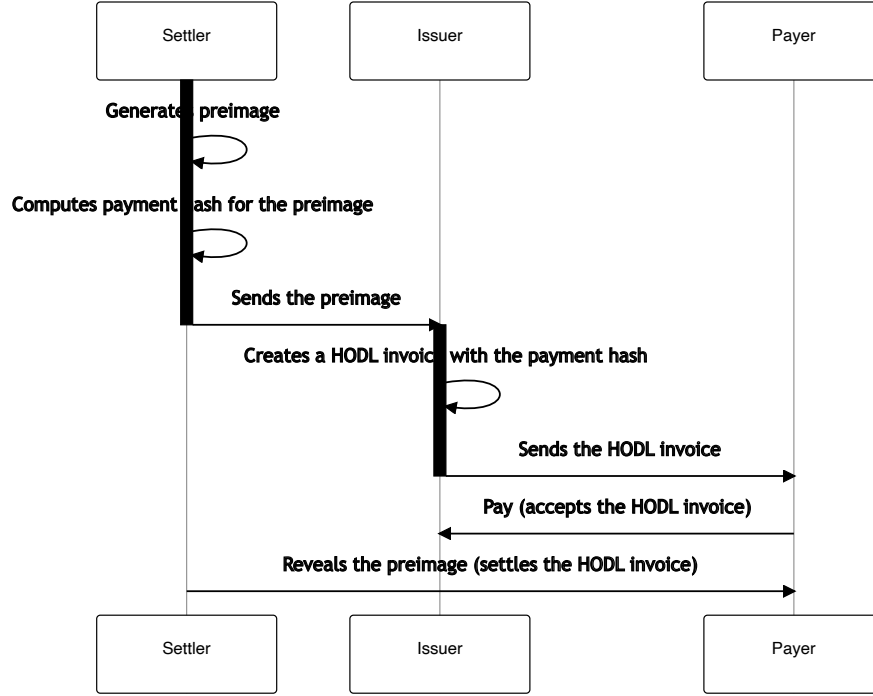


Figure 11: Lightning Network HODL Invoice Sequence

If one uses cryptographic keys as a preimage in the scheme described above, one can think about this scheme as being secure, atomic micropayments in exchange for cryptographic keys.

Having a message that is encrypted with a specific key, we can construct an invoice using the key as a preimage and compute its payment hash. To decode the message, the payee needs to pay the invoice and obtain the key (preimage).

### 3.8 Replying

The node (replier) that is happy to accept the broadcasted message instead of broadcasting it further is replying it back. It is done with ResponseFrame that is sent back to the node that was the sender of the topic.

The replyier\_certificate has two meanings here:

1. It allows the sender of the topic to identify that the gig contractor is a credible service provider by checking the certificate and verifying its "hard" certification (e.g. driving licence) with the specific certification authority being a trusted third party (e.g. government agency or specialised certification verifier)
2. It contains the replier public\_key that is used to verify the signature of the Reply Payload.

The main part of the reply is the message that is encrypted so it can be used only after the sender (customer) will pay the network. The encryption is done by the replier and its correctness is verified by the network while traveling back to the sender. This is done in the following way:

1. Replier encrypts the message with the sender public\_key that is a part of sender's Certificate in the RequestPayload
2. Replier generates reply\_invoice that can be viewed by the sender and must be used to pay for the service after the service is delivered. Sender is not obligated to pay it if decides not to use the replier. On the other hand the invoice is signed by the replier and the replier cannot change it after the sender accepts the service to be provided by the replier.
3. Replier generates a symmetric key that will be used as preimage for payment hash used in network invoices.
4. Replier performs symmetric encryption using the symmetric key
5. Replier computes payment hash for the generated preimage

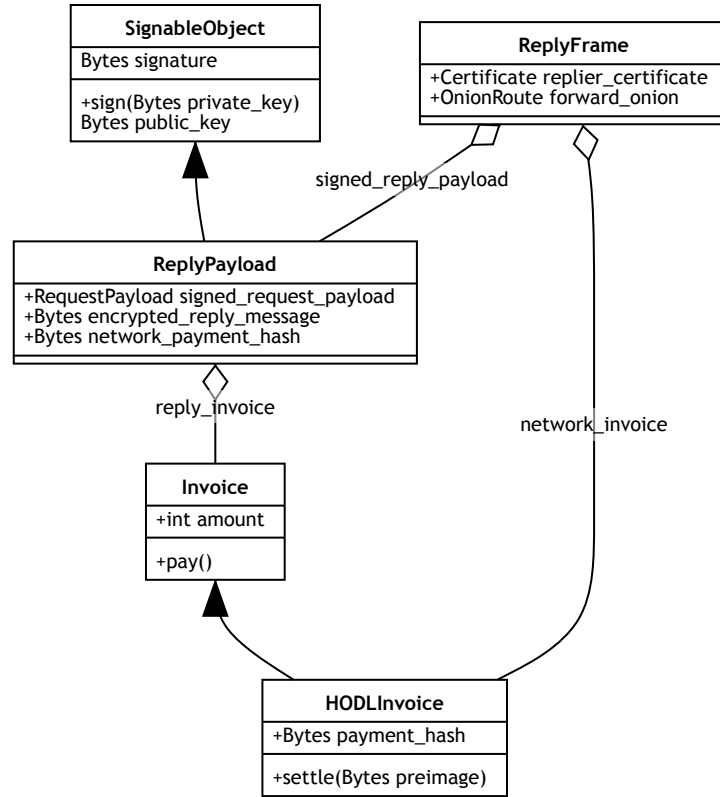


Figure 12: Reply Frame

6. The encrypted message needs the symmetric key and the private key of the sender to be decrypted, making obligator for the sender to obtain the preimage, and therefore the network invoice needs to be settled.
7. Request payload, encrypted message, network\_payment\_hash and reply\_invoice are signed with the private key of the replier, so its integrity can be verified with the replier Certificate

The ReplyFrame contains Replier Certificate, ReplyPayload and additionally:

- forward onion
- network invoice

When replying the forward onion is simply a copy of the backward onion, then each of the middlemen is peeling one layer of the onion, using its private key and sending it to the node that is found there in the peel.

Replier generates the original network invoice and puts there precomputed payment hash and starts listening on its state change. If the state of this invoice changes to Accepted it settles the invoice with known preimage. Then, while travelling back, each of the middleman adds its price on top of the invoiced one and replaces the network invoice with the own one and starts to listen on its Accepted state. Also it makes sure that the payment hash of the network invoice is the same as stored in the ReplyPayload. Once it reaches the sender and sender Accepts the network invoice the chain of acceptances reaches the replier. Once the replier settles its invoice the chain of settlement reaches the sender.

The sender retrieving the preimage uses it to decrypt the message.

If the one of settlements fails for some reason, it can break the preimage delivery process, therefore it is important to publish the preimage in an alternative way.

[This concludes the protocol.]

Attack	Sweet-Gossip defense
Spam	POW
DDoS	POW
Silent	Multibroadcast > 1
Chatterbox	POW
Sybil	POW
Eclipse	Multibroadcast >1
Censorship	-
Convert Flash	-

Table 2: Attacks

## 4 Discussion

### 4.1 Distributed Trust

Certification Authority hierarchy is used commonly in the modern internet design however it is based on the idea of trusted 3rd parties. Having Certification Authorities that coexist in a free market helps with decentralization of trust.

The other approach for distributed trust is based on the idea of a track record. Track record means that the specific customer (sender) or gig-worker (replier) were already involved in series of successful transactions. This can be implemented using proof of payments that can be tracked back to the specific service being delivered.

The other one is based on the idea of reputation (transitive recommendations), so trustworthy participants are recommending other participants putting their reputation on a table.

### 4.2 Mobile device connectivity issues

Holepunching [3]

Broken connection

### 4.3 Attacks

Here we are discussing common attacks that can be performed on the sweet-gossip network.

#### 4.3.1 The Silent Attack

The adversary tries to distort the distribution of the gossip nodes to cause propagation failure by failing to respond to the gossip protocol [2]. The same behaviour can be just a result of node failure that can happen naturally especially with mobile devices.

Multibroadcast > 1

#### 4.3.2 Chaterbox Attack

A malicious node retransmits repeatedly the same message [2].

#### 4.3.3 Sybil Attack

This is the most common form of attack in P2P networks, since creating large numbers of identities is generally cheap resource-wise, unless cryptographic puzzles are included as part of joining the system [8].

#### 4.3.4 Eclipse Attack

This attack can be carried out against a single victim or the whole network. The objective is to silence the victim by refusing to propagate messages from it or to distort its view by delaying message propagation towards it [8]. Here the attack can be performed against specific Certification Authority

### 4.3.5 Censorship Attack

Sybils seek to establish themselves in the mesh and propagate all messages except those published by the target peer. In contrast to the Eclipse Attack, in the Censorship Attack Sybils appear to behave properly from all vantage points, but hairpin-drop the victim's messages. The objective of the attacker is to censor the target and prevent its messages from reaching the rest of the network [8].

### 4.3.6 Covert Flash Attack

In the Covert Flash Attack, Sybils connect to the network but behave properly for some time in order to build up score. Then, they execute a coordinated attack whereby they stop propagating messages altogether in an attempt to completely disrupt the network. The attack is difficult to identify before the attackers turn malicious as they behave properly up to that point and build a good profile [8].

## 5 Supporting Services

### 5.1 Settler towers

### 5.2 KYC services as Certification Authorities

### 5.3 Push notification servers

### 5.4 Map servers

## 6 Applications

Sweet-Gossip Protocol is an enabler for building P2P apps.

## 7 Public Key

This is the Public Key (GPG) of Sonof Satoshi.

-----BEGIN PGP PUBLIC KEY BLOCK-----

```
mIOEY8pcpgEEAN+bUaEdg+y1WkdNc6U9LNkWB4ii0Neay4kUyU2NntHm1FAZPNSC
wxJ8P1brQn0GeeGnyfZtjZKTSn0Jor5YT4pHNlUBGFj3/BrihJCBRSJ878q02ct9
4RjXiNADVg1w3jRKRrk1CimmmL7VVK7oFZHd0311+8r/qIT4WN0ITydNABEBAAgO
MVNvbm9mIFNhdG9zaGkgPHNvbm9mLnNhdG9zaGlAZG9udHRydXN0dmVyaWZ5Lm9y
Zz6IOQQTAgA0xYhBIAALVutWo8Bqg5fDYtkf+QR5XMgBQJjylymAhsDBQsJCAcC
A1ICBhUKCgLAGWAgMBAh4HAheAAAJEItkf+QR5XMg2PYEAMCEB370PgxAAV+e
Kt4580PymI/rZOW06Cm9E6BqMdNqNx7d4udxbQYutkUr1xhLmLH1JTJwFhe3oMv
/3MUjm/VjIYrdnXAhHqvZA3502AyIWEQ660Q9whj57PY04YYcBZP/NDe4QuoUX9r
b3XzYIeJcQhUNG0zjjQJQ7bU7gcwuIOEY8pcpgEEAK/nkFTpi0iGtUI1RqWD46HA
nH7wTVXy2BVwHefRiDHz2hGgQiHXF6EU8mk9F2SVBj0TBNHGAWvXssT97Y8jiq6i
vJosx7VtolxBEDRL1PFMOH4whwu1rjDg8QR3KPkB3kMcXvD9ZHIB6FspVvhx2/Jk
V+PKLQ0ThhQITxFIKz4nABEBAAGItgYQAQgAIBYhBIAALVutWo8Bqg5fDYtkf+QR
5XMgBQJjylymAhsMAAkJEItkf+QR5XMgRngD/OGbCDFoL8hqppvuBuXBLHJVMLGh
fF/3fZyd1ZkjE+I1/LX5G/WSsLcAm/dmAVd8L1zat3PvdL57RHY06BEE4kdDEo8m
D1Z8SycI1yGaSS8dGCMaAFLz0xrJgER3NnXxg7BxCfREcUTawq1CE01QYx/71ib
GoTF/wPiCY/JQ1Ed
=Ei5q
```

-----END PGP PUBLIC KEY BLOCK-----

## References

- [1] A. Back. Hashcash - a denial of service counter-measure. 09 2002.
- [2] M. Burmester, T. V. Le, and A. Yasinsac. Adaptive gossip protocols: Managing security and redundancy in dense ad hoc networks. *Ad Hoc Networks*, 5(3):313–323, 2007.
- [3] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. *CoRR*, abs/cs/0603074, 2006.

- 
- [4] A. B. Johnston and D. C. Burnett. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Digital Codex LLC, St. Louis, MO, USA, 2012.
  - [5] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.
  - [6] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.
  - [7] S. Verma and W. Ooi. Controlling gossip protocol infection pattern using adaptive fanout, 2005.
  - [8] D. Vyzovitis, Y. Nopora, D. McCormick, D. Dias, and Y. Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and ETH2.0 networks. *CoRR*, abs/2007.02754, 2020.