haute école
neuchâtel berne jura
arc ingénierie
www.he-arc.ch

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Bachelor Project

HE-ARC 2016

Overclouds

Author:	*Supervisor:*	*Expert:*
Romain Claret	Marc Schaefer	Adrian Gschwend

July 29, 2016

Abstract

Overclouds is a project whose goal is to create an anonymous and decentralized internet data sharing service right through the browser.

Description

English

The initiative behind this project is to create a new generation of decentralized services to offer data sharing with a digital democracy over Internet. The services also want to be adapted to today's paranoia about Internet privacy as well as the preservation of knowledge for the next human generations.

The idea is to give the ability to the user to not rely on corporate servers, or farms of servers (cloud) anymore. With Overclouds, all users are now anonymous nodes connecting one to another freely and anonymously, with the goal to share and store data.

The network is a consensus of nodes. The data is moving from a node to another across the network via other nodes and ruled by the laws created by the community of users and accepted by consensus.

The project aims into the next generation of technologies to bring users a user-friendly interface without any kind of download and installation on their machine. To access the Overclouds service the only requirement it to use a standard browser and an Internet connection.

French

Le but de ce project est de créer des services décentralisés de nouvelle génération, afin de permettre un partage des données avec une démocratie numérique sur Internet. Ces services se veulent également adaptés à la paranoïa actuelle concernant la vie privée sur Internet ainsi qu'à la préservation des connaissances pour les prochaines générations de l'humanité.

L'idée est de donner à l'utilisateur la possibilité de ne plus compter sur des serveurs tiers, ou des fermes de serveurs (cloud) appartenant à des entreprises. Avec Overclouds, tous les utilisateurs sont des nœuds anonymes reliés les uns aux autres librement et anonymement, dans le but de partager et de stocker des données.

Le réseau est un consensus de noeuds. Les données sont en mouvement d'un noeud à un autre à travers le réseau via les autres nœuds et sont régies par les lois créées par la communauté des utilisateurs et acceptées par consensus.

Le projet vise les prochaines générations de technologies pour apporter aux utilisateurs une interface conviviale sans installation d'aucune sorte sur leur machine. Pour accéder aux services Overclouds, la seule condition est d'utiliser un navigateur standard et une connexion Internet.

Contents

1 Preface

1.1 Introduction

Today The world and more particularly the digital world has an important concern for privacy. Indeed, Edward Snowden's revelations on NSA's massive spying[36] led to a media scandal. People started to feel that their digital privacy was at stake by the worldwide "spies", either the government, companies, or even unknown threats yet. The result of this fear induces the Internet community to evolve and to think about a new economical and political world.

New behaviors emerged Right into the web users, a noticeable split has been made, into two classes of people. Individuals who do not know how to protect their privacy on the Internet, and people who do know how to protect their privacy. For the first group, some are not aware of the global privacy status; some don't care, and some don't know how to protect themselves. Concerning the second panel, we have some personal ethical problems, which led to this project.

The problem It is understood that people want to avoid advertising companies' and governments' tracking. It is also known that some people are scared that their stolen private data could be exposed publicly. However, some private data must be sometimes public and in some cases forced to be made public. From our humanistic and unpolitical point of view, we believe, that the data from terrorist groups (including their members or partisans) have to be denounced, made public and requires that the data gets locked down into a secured digital safe. Why archiving and not just deleting them? We, again, believe that regardless how bad the data is (as it is seen today), it is still part of our history, and it is our duty to preserve them as a legacy for the future generations to understand and learn from our present mistakes.

Intellectual Properties Read as IP. Our social and political evolution led to copyright laws, which prohibit the free sharing of any data such as art and knowledge by default. Indeed, our cultures specify that any intellectual discovery or advancement is by default protected by copyrights (we will not discuss the process here either the use of the copyright system). All this usually leads to knowledge or data retention, which makes us, the authors

of this project, feel that the IP system, is a serious threat for our humanity legacy.

Economy However, it is also understood that with our economic system, people need to make a living out of the IP, which leads into higher protection for their data. Our digital methods of protection are to encrypt the data with a digital key provided by the owner (generally companies). The encrypted data is then stored in data centers or on hard-drives (which could sometimes be bad for the long-term preservation). Owners then have the choice of sharing their (copyrighted) material and are usually also choosing to use encrypted manners to distribute them. We are also protecting the transmission to restrain people from looking at them without owners' consent and redistributing them with consent (money is in most of the cases involved). Therefore, the data is protected during storage and transmission, and we also have a layer above that allows only to decrypt the piece of data a user is looking at on the fly via DRM's technologies. Technically, the data is never totally in clear anymore.

Game of cat and mouse Some people are protecting data, other are trying to break the protection to retrieve the data in clear. Moreover, they both use encryption technologies to secure their transmissions. Meaning that the owners are protecting their data, and the *thieves* are also protecting the stolen data, to not be caught. In the end, the same data is being protected and transmitted, however, never in clear. This procedure makes us believe that something is wrong with the system and that it could be considered as a threat for the humanity global knowledge and culture.

Legacy We are confronted with another ethical problem. We are entering into an entirely encrypted data era, which will make all our data appear as a bunch of random noises, which we estimate on a personal point of view that for various reasons they are bad for us. We are thinking about how **all** our global knowledge is and will be sorted shortly. Data such as art or knowledge, how will our next generations of humans be able to retrieve it in a few years? How can we be sure that the unbreakable keys (quantum proof for example) that we are aiming to make today will not be lost and so the data with it? Now let's think about the idea of randomizing (adding noises) our communication signals that we are sending into outer space. The signals

that we are sending out to space are already today much different from what it looked like in the sixties. For example, purely analog signals (TV, FM radios) are depreciated and got replaced with a digital transmission (DVB, DAB), and of course, most of them are encrypted (pay-to-watch tv, pay-to-listen radios). Moreover, our communication technologies evolve into higher frequencies and always fewer consumption requirements. All this is resulting from the fact that today, Earth radiates probably much fewer waves than before, and it is seen as random noises.

Taking the wrong direction It is easily imaginable that soon a company will say: "Hey, we are specialized in transmitting encrypted data, nobody will ever know who you are, what you are doing, and when you are doing it. All we have to do is to buy from us, with a crypto-currency, a key every month.". See this as an evolution of the anonymity services such as VPN or TOR[102]. We do not have a sharp opinion on either it is a good or a bad thing. However, it will create a privatization of the data security, and it is most likely to destroy any hopes of retrieving the encrypted data in the future.

Overclouds The project aims into a different approach for protecting people's privacy and ensure a legacy for future generations. The main idea is that Overclouds is a consensus-driven anonymous network of nodes with storage and computation power. Each node is aware of other nodes, but they are unable by default to identify its owner on the main net (whatever the physical technology of communication at any given point of time). The consensus has limitless power over the network. It can, for example, decide to disclose the physical location of particular nodes, if they are considered malicious or wrong for the rest of the network (such as terrorists, scammers, black hats, and so on).

Nice to have We also would like Overclouds to be designed not act like random noises. Preferably, external and internal listener/watcher should see a nice mathematically driven signal. The mathemagics behind would allow any source node to predict exactly how the consensus will handle its data. However, it should be unpredictable for other nodes, but still elegant to look at.

1.2 Big Picture

After being introduced to the project, let us present our current vision for the outcome of Overclouds. Note that the results of the big picture are not the bachelor project, allocated man-hours are clearly not enough. And of course, due to the early stage of the project, the vision may evolve. The bachelor work aims to do research and produce parts of the proposed final project.

Global The goal is to provide a product easy to use, technophobe-friendly, that does not require any special software or process running in the background. We are focusing on the browser experience; any browser should be able to run a node and be part of the network. The user should be able to use any hardware and main net (see our current internet) connections to join the network, very low bandwidth, very low computational power, Hyperboria[56] support, or even be compatible with nonexistent technologies yet. The security allows to any user to be anonymous to ISP and censorship-resistant. Plus, any user could access its personal *desktop* with its data right from any browser connected to the main net.

The consensus A consensus-driven network with unlimited power over the entire network and putting the digital democracy to the top. It works in harmony with the newly introduced concepts of *Decentralized Storage*, *Data Tribunal*, *Mesh-of-Trust* and *Community Contracts* which glues nodes, users' identities, digital contracts, storage, gateways, and transactions recording.

Network A serverless mesh of self-aware and smart nodes. Every transaction is verified and is mathematically elegant. Intelligent enough for self-preservation, and bypass consensus decisions if they are judged as self-destruction (for example, if the new minimal computational power is higher than the current maximum available). Noting that privacy and security is being applied to any layer of the network. However, a consensus decision can always bypass any rule, even a privacy rule. A registry name could be available to access the network easily from browser's URL (.over?) without using a gateway like overclouds.ch.

Storage The network is using cooperative nodes to store and process computational work. Data has many states, private, public, time to live, a

countdown to go public, and versioning. Meaning that anonymous static and dynamic websites can also be hosted.

Crypto-Currency Consensus-driven currency, with an updatable genesis block by consensus. Nodes are automatically contributing to the network wealth. Coins are rewarded to identities by the consensus as a *proof of participation*. It represents how an identity *respects* and *contributes* to the network, which could give him leadership powers like a stronger weight for votes in specific cases determined by the consensus and contribute even more to the network.

Self updating An ultimate achievement would be a self-programmed network driven by the consensus.[45]

1.3 Objectives

As it is today, the data is centralized into specific operators like Google, Facebook, Dropbox, Tor hidden services, and much more (expect of services like Freenet[26]). They are usually using encrypted protocol to transmit data to their client but are most of the time stored in clear on their cloud servers. Sometimes the data is stored encrypted, but the operators must comply with the local laws of the data centers, which could compromise the security. The user could encrypt the data himself; however, technical knowledge is required and sometimes the rules of the storage operators does not allow homemade security due to laws they are submitted to. Note that during the data transfer, even by using SSL/TLS protocols, the IP address are visible. Solutions to IP anonymity are commonly provided by *trusted* VPNs, TOR[33], or darknet services like Freenet, I2P[30], etc. Also, it can be noted that encryption keys could be stolen if vulnerabilities are found like heartbleed[75]. Indeed, advanced security for anonymity requires some specific software to run, and could be discouraging or even dangerous for uneducated people. Plus, the more protection via encryption is used, the more data is *lost* for the human legacy. Indeed we cannot archive or index encrypted data.

The goal for this Bachelor Project is to try to answer the question:

Is it possible to provide a distributed data storage solution that prevents spying, and that simultaneously doesn't open a Pandora box for illegal and unethical data exchange?

The foundations for this project have been done during the Spring Project[24], which goal was to prepare the material and planning for the Bachelor Project. The work done during the cited project was mainly doing literature research and prototypes for browser only communication, encryption, and consensus.

1.4 Specifications

The spring project aims to study the state of the art, provide solutions and achieve incremental prototypes that meet part or all of the following objectives. It is of course not required to reinvent the wheel.

- Study of the state of the art
- Sharing and transfer of data ownership
- Data decentralization (distributed storage), which is encrypted by the network via cooperating nodes
- Pseudo or full anonymity from the ISP
- Rating and aging notion of the trust level from network's nodes via a mesh-of-trust.
- Banishment of data and node certificates from the network (via a Data Tribunal)
- Private keys theft resistance, even if a heartbleed[75] type vulnerability is discovered on the nodes later on [96]

1.5 Management

An iterative approach has been used during the progress of the project. So, the SCRUM methodology was chosen. The Backlog and the Sprint advances can be found in the annexes.

The project planning has been done for eight weeks. It was divided into eight sprints starting and ending at each week Wednesday. We will be using the software called Vivifyscrum to digitalize the scrum process; the school will be paying the 10$ fee for each month of the project.

A global replanning was made during the second sprint because objectives weren't clearly defined and the tasks were too detailed (and maybe unrealistic on the long term).

At the end of Sprint 3, a replanning has been made due to multiple problems with the Ethereum network. It was decided to focus on the decentralized storage.

As the project ended, all the weekly objectives had been respected expect for the consensus related implementations.

Based on the student's log, the time requirement for the project of 340h has been made with 341.25h. The R&D itself as produced 289.50h:

- Research: 149.00h
- Development: 140.50h

We can note that starting during development, the student had usually forgotten to use the *Vivifyscrum*, however, the student log was kept updated.

2 Analyses

2.1 Blockchains

2.1.1 Worth it?

It is important to note that with incoming quantum computers (predicted to appear wildly in about twenty years), the mining structure, the security and the anonymity must change from today's perspectives. As for today, the blockchain technology is at its hype, meaning that we see it as the best thing in the world.

However, from now the hype will decrease and maybe a new technology will emerge or the blockchains technology will evolve or be modeled to go in a direction we do not expect yet.

Yes, from today's perspective, the timeframe is pretty significant, it is worth the interest.

2.1.2 What's next in crypto-currency?

As of actually, considering that the technology of blockchains will not change, and is still in use for crypto-currency, it could take two types of path.

One of the paths is the neverending death and birth of crypto-currencies. Indeed, once the mining is no more profitable, the security sharply decreases because miners are verifying the transactions and are playing the role of consensus for validating transactions. Miners are mining as long as the devices allow a profit (power consumption, device rentability, etc.). Best case scenario, the hardware technology continues to involve, as well as the required computational power. (Note that we are currently brute-forcing the solutions.) Moreover, based on the model of crypto-currency of Satoshi Nakamoto, Bitcoin, at some point in time, the maximum amount of coins will be reached, and the network will not generate coins (rewards) anymore. At this point, the only income of the miners will be the transaction fees. If the transactions fees are not high enough to motivate the miners to continue mining (and verifying/validation the operations), the currency will die due to the lack of security. So the miners will move to new profitable crypto-currencies (note that they could have an advanced hardware for mining at this point, which will help them to start pretty well).

The second path is the modification of the source code of the actual crypto-currencies to make it compliant with the market evolution. For ex-

ample, increase the maximum amount of coins, or make public keys quantum proof. Indeed, currently, the **ECDSA** is not quantum proof (however the hashing is at the moment, but SHA3 is ready, just to be safe). The problem is ECDSA, which during a transaction sends the public key, and theoretically, a quantum computer can guess the private key from it. However, the address is still secure because it is the hashed public key.

However, the second path is **killing** the concept of a stable currency based on expendable raw material stock, and the social and economical results are pretty hard to define. A secondary question would be: What will happen, if tomorrow, we find a new gold mine, which holds the same amount of gold already retrieved (doubling by this mean the maximum quantity of gold available), and with a retrievable difficulty level a lot decreased, so it is again profitable to mine?

So, we do not know if it will be a next big crypto-currency. In my opinion, I would bet on Ethereum. However, again, it is personal.

2.1.3 Predicted evolution in blockchains

This subsection will be pretty short because at the moment, this technology is only starting to go down on the hype slope, and the only real evolution that pops out recently is the first version of **Ethereum**[105] (2013a) and more recently (2016) the Homestead version of Ethereum [34].

The particularity of Ethereum is that it uses the currency as fuel to run smart contracts on the EVM (Ethereum Virtual Machine) using the power of each node on the network to do a calculation, and creating a consensus on the output. This technologies evolution has for example created a startup company named *Slock.it* and an alternative "currency" *DAO* (which is unmineable) that allows the IOT (internet of things) to interact with the crypto-currency Ether. It allows, for example, to control a lock, in a hotel, a door could be locked until a client paid the door to open.

2.1.4 Proof-of-Work

Read as PoW[35, 63]. It is a protocol aiming to reduce the risks of DDOS attacks and family abuses by requiring that the client has done some computational work (processing time) before sending a request. It was a solution developed mainly for our financial world of transactions.

2.1.5 Proof-of-Stake

What is a stake? It is globally something that holds. In our case, it is more like a flag that can keep a land (a claimed property).

Proof-of-Stake?[66] Read as PoS. Usually, in blockchains PoW, miners validate the transactions that came first depending on their CPU power. Note that the more CPU power you have (GPU, FPGA, ASIC, etc.) the larger is your influence. POS is the same thing but with different paradigms:

In one of them, Stakeholders validate with something they own (raw material as an internal currency). Moreover, in a simple manner, everybody has a certain chance (proportional to the account's balance) per amount of time of generating a valid raw material.

Another paradigm would state that we are not working with the amount of raw material owned, but with their age (for example, the raw material is multiplied by the time that it was unused) which gives a weighting factor. However, with this paradigm, a collusion attack is pretty important, because we could have a super linearity by accumulating aged raw materials.

There are other different types of approaches but we will not details them all because they are not the best of consensus algorithms. (elitism, identity, excellence, storage, bandwidth, hash power, etc.)

Now, on the security side By using raw material (sort of digital assets) defined by the consensus PoS avoids a Sybil[44] attack. Which is a technique where the attacker is trying to compromise a system by creating multiple duplicate or false identities. It is resulting into including false information, which as a result can mislead the system into making not intended decisions in attackers' favor. By the way, PoW protects itself against a Sybil attack by using computational resources that exist in an extra protocol.

However, in PoS' traditional approach, we have two major problems. The first is Nothing-at-Stake, and the second are Long range attacks.

Nothing-at-Stake The dominant problem is that smart nodes have no discouragement from being Byzantine[68]. Indeed, signatures are very easy to produce, and they will not lose any tokens for being Byzantine. Another problem is that nodes with digital assets could never spend.

A solution to this would be to have a security layer on deposits, which would cancel Byzantine deposits. To achieve this, we would need to store

information about nodes and their immoral behaviors (which are decided by the consensus) so the consensus would be able to punish them. Now, this works only if the transactions are not hidden (with the proof of malicious actions). Also, we should note that this security layer would ask more power for the consensus during the use of punished accounts. Slowing down the consensus is not acceptable because it acts as the authority and by this mean should be the cheaper to operate in power. Punishing the attackers with power consumption is fair.

Compared to PoW, where attackers are not receiving compensations for their computational power (which is a disincentive). The PoS' security layer is trying to disincentive attackers by removing their digital assets. It could be an interesting social experiment, however, in our human's economic point of view, attackers should be well disincentivized.

Long Range Attack In this type of attacks, the attacker controls account with no digital assets and is using them to create competing version of transactions. This attack is touching both traditional PoS, and the deposit security layer (as long as authentication ends in the genesis block).

The solution here would be to force nodes (and clients) to authenticate the consensus (for example with its state) by signing with the nodes that have something at stake currently, and nodes must have an updated list of nodes with deposits. It is usually called the *weak subjectivity* method.

Ghost From the full name, Greedy Heaviest-Observed Sub-Tree protocol, introduced by Yonatan Sompolinsky and Aviv Zohar[94], it allows the PoW consensus to work with much lower latency than in the blockchain protocol from Satoshi Nakamoto [88], and of course keeping it secure. Indeed, in blockchain based PoW, a miner is rewarded for each block found so the other miners can continue to mine on top of it. However, when a miner produces an orphaned block (a block that exists in the chain), they are not rewarded for their work, plus the consumed power was in vain because the work is unused by the consensus.

Here comes the solution, Ghost, which includes orphaned blocks. It introduces the notion of rewarding orphaned blocks to miners and increasing the security of the consensus with increased validations of a block in the blockchain.

Casper Now there is a friendly ghost in town; it is Casper[22]. This protocol is based on Ghost, and must be integrated into Ethereum for the Serenity[21] version (final), however, the Metropolis version must go out before. We should also note that they released the Homestead version on 14th March 2016 (about a month before this Report release). It will work on the smart contracts.

Finally In comparison to PoW, PoS is much cheaper to secure, transactions speed is greater, and it is maybe the stepping stone into scaling the blockchain technology.

2.1.6 Attacks on blockchains

Blockchains is designed to be controlled by the consensus of nodes. It means that it can not be owned not controlled by a third party. Until now this goal has been achieved. However, experiences showed that the system is not perfect.

The 51% attack Most interesting way (in a social experiment point of view) of rewarding for the attackers. Indeed, if the attacker controls at least 51% of the consensus, it is possible to manipulate transaction by validating malicious transactions. Pools owners can do this. Note that as it is today (for actives crypto-currencies), it is no more possible to mine on your own and be profitable, miners are forced to join pools and distribute the work and rewards between them. Meaning that it creates a vicious circle, the more miners are in a pool, the more power it has. The more power it has, the more reward are generated. Finally, this results in attracting, even more, miners because they also want a bigger and easier reward for mining, which leads to the security risk of malicious pool chief who will control the currency and the transactions. All around the internet people are always saying that it is dangerous, in fact, they are asking others to stop making a profit for the good of others, which is a selfish human thinking. Can't wait to see it as a case of a figure.

Spam attacks The idea here to make many transactions to the victim's wallet (by its addresses) and paralyze the legit transactions and by this way its incomes. Indeed, the network will have to process all the spam transactions as well as the legit transactions, meaning that the delay is added before

receiving the legit transactions. In some cases, like for Wikileaks[101], which is depending on this kind of funding to live, it is pretty bad. Plus, since many transactions appear in a block, its value increase and miners will jump on it to get the reward, meaning that the legit transactions are a bit behind because they have a lower reward. However, usually, the current crypto-currencies have an anti-spam solution. They have a minimum fee, and they increase the fee after each new transactions.

DDOS on exchange platforms The profit behind this type of attacks is to either steal wallets or ask a reason to release the servers. The crypto-currency is only affected by the depreciation of its value in "real" money because they are not able to trade, and they are more likely to switch to another exchange platform or currency.

Special dedication to Mining malwares It is funny to see that hacking is evolving with the hype. Instead of having zombie computers doing nothing waiting for DDOS attacks or whatever they are used to, they are now mining coins (generally connected to a pool). How smart is that? We think that it is amazing!

2.2 Proof of Activity

The protocol from Bentov, Lee, Mizrahi and Rosenfeld [15] which is implemented into PeerCoin (and its clones), is considered as a hybrid of the PoW and PoS. The nodes are doing PoW work by mining blocks and at the same time with the PoS (meaning that the blockchain includes both types of blocks).

The procedure

- The PoW miner mine.
- Block is found, the network is notified and creates a template. (multiple templates are possible)
- The block hash is used to find random owners by using its hash as numbers to determine owners (nodes from the network).
- Turn by turn each chosen owners sign the key with the key of the block.
  - If a chosen owner is unavailable, the process paused. (it is not a problem this concurrently miners are still mining and generating new templates with different owners)
- At some point in time, blocks will be signed, and the reward will be given to the miner and the owners.

Continues data exchange To reduce the data traffic, each template does not include a transaction list during the signing process itself; it is the last owner (signer) that is adding it when creating the block.

2.3 Communication

Considered to be the most important part of the project, without internodes communication, Overclouds has no meaning. We are also aiming to work only on a browser only system, leading into looking at modern technologies written in Web-driven languages.

Propagation We began to look at homemade solutions. In the cases of communication by data propagation, the path optimization will not be possible, and the global status of the network will be tough to maintain. Plus, it will not be easy to be sure that the chunk arrived at the destination. However, it is easy to push data into the void. Also, note, that a protocol must be made to avoid to the propagation of the source address and preserve anonymity.

3rd parties The most common way to do peer-to-peer from the browser is to relay on a 3rd party service to do the signaling and put peers into relation. To preserve the anonymity and security integrity, we must, in this case, maintain a list of *trusted* nodes (gateways) by the consensus to join the network. We could use TOR technology, Figure 20 in the annexes, but a browser only integration is not done yet. The 3rd party services could be either servers running custom code, for example, wrote in NodeJS, either companies specialized into signaling services like peer5[82], openpeer[54], peerjs[97], etc. In the second case, we must give even more trust into their services (hosting and signaling), we are hardly favorable on this option, how can we ensure their services will not be shut down or will not censor our project because they do not like it or if they are asked to do so?

Organization's Server If the 3rd party servers solution is selected. It could be interesting to create a protocol that gives the consensus the control over the servers and manage the costs. Indeed, a consensus-driven payment via crypto-currencies directly to the 3rd party host could be interesting, but it would mean that the consensus owns capitalized crypto-currencies. We could imagine that the network would mine crypto-currencies.

2.3.1 WebRTC

Pushed by HTML5 standards and companies like Google, Mozilla, Opera, etc. WebRTC[48] provides a clear API, optimized for Real-Time Communications (RTC), in an always growing developer community, and the major browsers supported (Chrome, Firefox, and Opera). Note the Chrome and Firefox have interoperability. However, a server is needed for the signaling (sort of tracker).

Datachannel[85] Based on the top of the basic WebRTC, and almost real-time, the main limitation is peers' speed connection and Javascript itself. It has two modes, **reliable mode**, which provides TCP-like guaranteed packets' order and no loss. Moreover, the **unreliable mode** which provides like UDP no guarantees for the data transfer. It allows encrypted communication and data transfer between compatible browsers (Chrome and Firefox). However, threats still exist, such as human activity. Indeed, a person can send malicious data like a virus to other peers. Note that the transfer is said to be secure at the moment, but is not unbreakable. It is recommended to encrypt the data before the transfer. In Overclouds, we would encrypt each chunk. Like the basic WebRTC, linking peers is done by signaling servers. The figure 24 from the annexes schematize an advanced procedure to make a data channel between two clients, which should prevent censorship or freaky firewalls.

ORTC On the other side, Object real-time communications technology is pushed by W3C & Microsoft. It is younger than WebRTC, 2011 vs. 2014, the main difference[91] between the two protocols is the supported media protocols and also the fact that ORTC does not have a standard for its signaling protocol. It is not clear today, what major differences will be between them in the future.

2.3.2 What about Overclouds?

By taking the pros and the cons of the previous communication solutions, our choice goes to **WebRTC Datachannel** which is the closest technology to our needs. However, the signaling requirement was challenging. One solution would be to have a trusted signaling node list controlled by the consensus.

Signaling free The main problem we were confronted to while doing research on WebRTC, was the incompatibility with Overclouds' vision of a browser only solution, which means that the user should not install any software expect a browser to be able to connect to the network. We finally found a solution to use the very promising WebRTC, however, the downside is the signaling procedure, which centralizes the tracking of nodes on the network, but it could be accepted if the server is trusted and maintained by the consensus. To our greatest pleasure, we found a solution to make the signaling procedure serverless. Users will still need to communicate their secret peer address somehow, but the addresses will not be managed via a 3rd party. The next step will be to find a procedure to transmit the peer address securely. The figure 25 from the annexes schematize how the hole is made in the firewall with STUN servers.

2.4 Cryptography

Privacy Being an integral part of Overclouds, research has been made to find the best type of client-side cryptography (right from the browser as the highest priority). We were looking for a fair middle ground between performance and security.

2.4.1 From Scratch VS Libraries

1st Question Is it possible and does it exists right from the browser?

We started looking at what is done in JavaScript and we found an interesting list of *premium* libraries (maintained by prestigious organizations) such as Stanford Javascript Crypto Library[95], MDN[73], W3C[93], Google Closure[49], or msrCrypto[76].

Then we looked at other crypto libraries such as forge[32] (a native implementation of TLS in Javascript and tools designed for crypto-based and network-heavy web apps), jsHashes[64], crypto-browserify[98], etc...

2nd Question The natural question that followed was: is it worth make it ourselves?

The answer came pretty quick while navigating across numerous forums. It is a pretty bad idea if we do not have a team dedicated to it and an active community to test it out. Even large enterprises such as Microsoft or Google are struggling a bit on the last part.

However, for the fun of it, we looked at solutions to start a homemade cryptography library. We found two interesting potential starting points to make it work with the browser, a Symmetric Encryption sample [61], or a procedure for Digital Signatures[60].

Decision Shortly after the second question, it was clear that it will not be possible to create our cryptography library in the time given. So we decided to find the *best* library out there for our project.

2.4.2 Comparison of some JavaScript Cryptography Libraries

Based on the table 3 from the annexes and the following referenced tables we can notice that **sjcl**[95], **crypto browserify**[98], and **forge**[32] algorithms have been optimized for defined objectives.

Also see tables 1 and 2 and their related figures 29, 30, 31, 32, 37, 38
Read the table as going from Amazing to Awful.

2.4.3 A killer

After taking time doing research about the above algorithms, we came across a pretty amazing algorithm based on Sha3: **BLAKE2**[50, 87].

BLAKE2 outperforms MD5, SHA-1, SHA-2, and SHA-3 on recent Intel CPUs and it has **no known** security issues. Plus SHA-1, MD5, and SHA-512 are susceptible to length-extension.

It is a *new* algorithm designed specifically for **performance** and is multifaceted **BLAKE2s** (optimized for 8to32-bit) and **BLAKE2b** (optimized for 64-bit)

2.4.4 Now what

If we look at the figure 39 from the annexes, BLAKE2 is dominating the two best above. **rusha** is close behind it, and forge's *sha256*. Plus we can note that the curves display a nearly completely linear performance.

Also on at the table 4 from the annexes with the figure 40 from the annexes, we notice that BLAKE2 is in its own category.

2.4.5 What about OverClouds

We can note that we are not interested in SHA1, because of potential security flaws. SHA256 is much better for us. However, BLACK2 is pretty amazing for the hashing. We will try to make it work in the following implementation.

Now, if it is not functioning for whatever reason, we will certainly go with forge, crypto-browserify, or sjcl. The problem with forge and crypto-browserify is that we must trust a company or an individual. With sjcl however, we are confident into a scholar institution.

2.4.6 Special Mention

We researched also PGP[83] and GPG[67] technology. We conclude that it is still a good encryption protocol, but too heavy for Overclouds.

2.5 Consensus

Read this subsection as a teaser from the final project analysis on the consensus concept. Indeed, the time allocated for this research and analyze was null. However, knowledge grows over time even if the research was targeting something else.

2.5.1 Consensus != Blockchains

The blockchains technology being very popular nowadays, it sometimes can put eye cups on our field of decisions.

Blockchains have indeed proved that it works as a consensus. However, a consensus with highly fault tolerant networks and overcome the Byzantine (Two Generals Problem) is not something that only blockchains have.

For example, Maidsafe[71] uses another technology to obtain a consensus[79]. Instead of using the whole network to validate a transaction, they give the consensus role to a random group of nodes.

Comparing They both have pros and cons of course.

- Blockchains
  - Pros: Shared global record of all transactions.
  - Cons: The chain can be gigantic (Bitcoin more than 60GB at the moment), and the file must be synced between all network's 6000 plus. Network speed. nodes[11].
- MaidSafe
  - Pros: Bandwidth speed limitations only. Low data storage consumption.
  - Cons: Small groups of nodes are playing the role of consensus for transactions. Nodes could never be aware of transactions that happened elsewhere if they are not related to them at any point in time.

2.5.2 Overclouds Consensus Concept

Blockchain This technology could become heavy. As a solution to decrease the long-term size, the consensus could agree on generating a new genesis block and distribute it to the clients.

Ethereum Blockchain solution with smart contracts capabilities. Custom Genesis Block out of the box for private chains. Storage is not possible right from the ethereum itself. However, a clever mix of ethereum implementation and decentralized web storage (webrtc, webtorrent, etc.) could work.

Maidsafe See it as a small group consensus. It includes a decentralized storage, but it is not open to developers yet. Plus the hype is currently much lower compared to Ethereum based on Google trending. In the end, Maidsafe doesn't even come close to ethereum's community and adoption progression.

Alternative blockchains None found looked as attractive as ethereum. Indeed, it is the only one giving a virtual machine to run code on the network (smart contracts).

Browser based blockchain It doesn't exist. Browser solutions to mine exist, however, a combination of JS, HTML5, WebGL. Investigating in the future could be interesting.

The one The chosen consensus is Ethereum. A new architecture with an ethereum private blockchain has been made, which can be found on the figure 21 from the annexes.

Development NodeJS+NPM are at start designed to work on a server. However, with the browserify technology, nodejs code can be a bundle to run in the browser, but it's not a universal or magic technic, it could need some work is the npm package, for example, isn't designed to operate with browserify right out of the box. EthereumJS has already modules compatible with Broswerify.

2.6 Storage

A lot of work and effort has been in the last few year on the decentralized storage. In our case, storage is controlled by the consensus; storage allows the identities to save data into the distributed nodes of the network. Integrity checks are done by the consensus as multiple nodes are doing the same computation work and chunks storage including redundancy. The storage is divided into two parts, the data stored by the users and the data generated and manipulated by contracts. Contracts' data if composed of the genesis block, keys, and identities info. The figures 6, 7, 8, 9 from the annexes relate to the analysis.

Data Stored across the network, it is spread on the network as encrypted chunks belonging to an identity, the private and public keys for the data are spread across the network and belongs to the network as well. Each chunk has redundancy which is also spread across the network.

Verbs Only Read, Write and Destroy makes current sense in our latest architecture. Indeed, Edit has an only sense if data versioning is integrated, which has no use currently since the shared data have a unique hash. To think in term of versioning, each version has its hash. We could work with the data addresses to keep a consistency and a low network load for Readers.

Flags Used to filters, and to ban, they must be persistent in the file, identity.

Bans Data and identities can be banned from the network after a trial or directly from the consensus. A flag, seen globally by the network, is raised for the banished content or identities. During a trial, a random amount of random identities is asked to vote on the event that led to the trial, with the help from the comments left by the blamers.

Flags Node can flag specific certificates or nodes on the network and apply rules on them, such as filters for minimum certification level required to be able to connect to them as a relay. The network then uses the flags, and route communications depending on the nodes' preferences.

For example, in the case of a node receiving data from another node that is not matching its filter, the data would be rejected at entrance. In the event of spam, which assumes that the attacker knows what destination (that he sees as void) to target, the target node will indeed consume power to deny requests. A solution has to be found for a case of a figure and prevent spam.

Another example, the node rerouting. A node can decide not to relay data. In this case, the data is either sent back, which can result in a load problem. Either the data is lost, which is in the event of a UDP-like protocol bad. In the case of a TCP-like protocol, the node would be searching for another node to go through. The second option could on another hand surcharge by exploring new paths.

Blames An identity can anonymously blame chunks or identities and leave a reason for the blame from the list of multilingual generic reasons. Note that the consensus can also create new categories. Each node can blame a specific data only once. A blaming identity is then linked to the note used to blame, by this mean it cannot vote more than once per data. Plus the first identity to use the node gets the blame validated.

Blames are telling the network that an identity or a chunk is not appropriate to the other nodes. Multiple blames on a node may result in a ban of its certificate. Multiples blame on a particular chunk or chunks belonging to the same data resumes into a revocation of all rights given to nodes even the owner.

Global knowledge Blockchains technology could be interesting for this purpose, however, we concluded that it is not the only possibility. Indeed, technologies such as VCS can be compatible also like directories (free track of a list of nodes, activities, etc.) or versioning (pull requests, etc.).

P2P While looking at innovative solutions to distribute data across the clients connected to the network, the peer to peer solution got retained. More precisely the bittorrent protocol looked promising for the needs of the project. Luckily a javascript and webrtc implementation of the bittorrent protocol named WebTorrent[103] has been found and is working right from the browser.

Data owners The owning rights are given and managed by the network to a particular identity. The owner can give as he wishes the reading, writing, and executing rights to any nodes or identity on the network. He can also set a public access for trust levels. The ownership is, of course, revocable or transferable by the consensus. However, the owner can also transfer the property to another identity. Both parties must accept the transfer. Note that he cannot transfer blamed data.

2.6.1 Content Management

Based on the Webgate and Storage architecture, the content management provides the concepts and features for Overclouds' decentralized storage. The figure 16 from the annexes shows the concept.

2.6.2 Data Tribunal

Providing a democratic authority for the data stored on the network. Allows to blame and ban data and users. The ban hammer also provides a digital tribunal with the selection of random judges during a trial. Once the required amount of blames reached the network, select not related random identities and asks them to rate the blame. The digital judges would be able to read the reasons left during the blaming phase. However, they are not able to read the content of the data itself. Except if is the consensus decides otherwise. Owners can ask for a second trial if they think the judgment was unfair and add a generic argument. For the second trial, different unrelated random nodes will have to rate the blame again. The decision from the second trial is definitive. The data is either archived, either the owner and related nodes get their rights back. The figure 14 from the annexes relates to the data tribunal concept.

2.6.3 Data Sharing

The content is distributed differently depending on user's willing to use an identity or not. In the case of anonymity, the shared data is only active as long as the user is connected to the network or that an identity decides to add it to its account. For identities, they have the choice to save it to their account, and by this mean retrieve the data during another session.

2.6.4 Existing solutions

They are mainly all requiring a client to work, they also most of the time provide gateways to connect to a server running a client from the browser.

Swarm[38] It's a smart contract which provides a decentralized storage using the power of ethereum, but it's not convincing at the moment. It has potential, however. It comes close to what overclouds wants to do, but it's expensive (in currency value) while no scaling is done on ethereum to interact with the data. Indeed, an interaction (read/modify) costs an amount of Ether.

Filecoin[9] is a project maintained by the creators of ipfs, however, it's not open yet. The paper is interesting and promising.

SyncNet[77] is a browser with decentralized content. The approach here is interesting because the client is right into a custom browser, which serves only decentralized content.

Storj[109] is a decentralized service based on their own crypto-currency. Users are paying or paid for using or giving storage to the network. They are like AWS or Google Storage companies. They are monetizing the storage.

Symform is a dropbox like decentralized storage. Users give storage to obtain storage; it's based on a win-win concept. The data is chunked and distributed into across the network. However the DHT is owned by the company, and they do all the dirty work for the redundancy control, etc.. Of course they resell storage.

Bittorrent[39] is a good old protocol that is well known for proving the reliability. They are by default chunking the data, and all current clients are encrypting the communications between peers. However, the data is sent in a whole, not only parts and spread across the network.

Webtorrent[92] is a javascript implementation working right from the browser. The webtorrent client (browsers) can talk directly to each other via webRTC. Gateway client which can talk bittorrent and webtorrent exists

as well and allows a bridge between bittorrent (UDP/TCP) and webtorrent (webrtc).

IPFS[14] It is really interesting but no protection is done right at base level. They are mainly providing a client that allows unduplicated data across the network. As it's only working at base level, many layers must be put on the top of it. A browserify version exists but isn't friendly at the moment; it's a bunch of APIs. The files are also chunked by default (if the size is bigger than the threshold). We would have to implement the layer of encryption with keys and forcing redundancy. And provide a browser-only solution.

2.6.5 Overclouds Storage solution

In our case, we are looking for a solution that is a browser only, data encrypted, keys distributed, with redundancy, spread across the network. We will be using webtorrent and implementing the layer of encryption with keys and forcing redundancy. But we start with a proven to work browser client. We talked over Gitter.im with developers of IPFS and WebTorrent for more information about the chunking of files. Webtorrent responded that it works exactly like the bittorrent protocol. If a peer is not providing data, it's excluded from the list, the client may then choose only to share some files of a whole downloaded folder.

2.7 Global Architecture

During the spring project, where this project starts from, we bootstrapped global solutions and concepts, resulting into a first comprehensive overview of the project. However note that during the bachelor project new research and analyses have been made, and the project has evolved, which resulted in new architectures, and new recommendations. The spring session version of the global architecture can be found on figure 2 in the annexes. Of course, this architecture is not definitive but contributes to the understanding of what we are trying to do. The latest global architecture version is found on figure 1 in the annexes.

The main point of this architecture is to create a consensus-driven with a decentralized data storage network.

2.7.1 Consensus-driven

The goal is to apply the rules made by the identities. Overclouds control the security between the nodes. Indeed, from client's point of view, they are talking about the network and not a particular node from the network. The network work is split into encrypted and distributed chunks to nodes. Note that the chunks are encrypted for a particular node as the keys are stored and owned by the consensus. Each node is participating by default to the network storage and computation power. The trust brainstorming can be found in the annexes on figure 11

2.7.2 User

Considered as someone willing to access the Overclouds network. As an entity, the user can at least create one identity which is defined by a Pair of Public & Private Keys and Coins. Then at each connection to the network, the user must log in with the private key to the public key which works as the username. This process provides an anonymization for the users.

2.7.3 Identities

Users can use any node from the network. For a user to identify himself, he needs to generate at least one unique identity. Identities are created from an existing node that is already part of the network. An identity cannot

be alternated. The user must generate a new one if the previous one was banned, and restart the process of acquiring reputation from the network.

2.7.4 Node

See as a peer for the network which runs default a virtual machine (which executes networks contracts) and provides computational work to verify network transactions.

2.7.5 OC Blockchain

Based on Ethereum[105], Overclouds is running a private blockchain. It provides a history of transactions, smart-contracts support, and network activity tracking.

2.7.6 Gateway

A browser-only solution allowing a node to join the network. A user can choose to use an Official (maintained by the Developers), Community (managed by a community of users) or Private gateway.

2.7.7 OC Contract

Working as the main unalterable contract of the network. Its goal is to provide the basis for the network to work. It's applying the rules made by the community. The communication between the nodes are all passing by this entity, indeed, from client's point of view, they are talking directly to the network and not a particular node from the network.

Democracy Identities can create votes and of course votes for submitted proposals, which are creations or modification of the rules governing every transaction on the network. Each identity is weighted by default to the initial vote unit and can vote once. However, the consensus can decide to give more weight or votes to distinct identities. By default, nodes can be used only once to give a vote (except if the consensus agrees else-wise).

Executing User are not interacting with the storage themselves, they must use request the network to do the work such as read, write or execute.

Internal crypto-currency Nodes are automatically retrieving units of the internal crypto-currency with their proof-of-activity. The network owns the currency. Rewards are given to identities for good behavior and participation. The coins are transferable to unbanned identities. However, an identity has no interaction access to its balance during the trial phase. If the identity is banned, its currency balance is returned to the network. The use of the crypto-currency is not clear for the moment, but it could be employed as a fuel like on Ethereum[37].

Reputation Using the internal crypto-currency as unit. While contributing to the network, identities are earning coins. Those coins are used to interact with the network itself. For example, identity gains or loses coins for being correct or incorrect on a blame after a data trial. A concept for the storage reputation points is found on figure 5 in the annexes.

Mesh of Trust Authority in charge of linking the proof of participation to the identities.

2.7.8 Community Contracts

Those are the contracts submitted by the community. It could be rules, third party storage and gateways. The community is voting for the contracts (rules) to include, exclude from the network.

2.7.9 Proof of Participation

New concept coming from the need for the network contributors. Identities with accepted contracts are considered as special identities, indeed, if the community accepts their contracts it means that they understood the community needs. They have a higher vote weight and are earning more coins.

Network requirements A node does not need an identity to be able to connect to the network. It connects automatically and integrates the mesh of nodes by giving storage and computational power. However, an identity requires a node to connect to the network and interact with it. So, a node works without an identity, but an identity always needs a node.

Security When everything goes by plan it's awesome, the best would be to plan also what is less awesome.

Internet Service Providers They should not be able to interpret the network activity. They are only aware of encrypted tunnels made to random nodes. Like the Tor network, the gateway nodes never the same. It allows a censorship protection.

Backdoor The project is not friendly for third party authorities like governments, police investigations, companies, etc. However, the consensus has virtually unlimited power over the network and can model its democracy as it is pleased. We can easily imagine that the consensus decides to disclose all the pedophiles from the network with the goal that the *real world* would punish them. The consensus decides what is right or wrong, and morality.

3rd parties Depending on the technologies used on the project, we could, in a first time, use third party services like Tor, Github, Twitter, BitTorrent trackers, etc. (note the public status and censorship of the example). The security could be compromised during the bootstrap phase in some cases because the ISP and others services could target and track nodes' activities.

Compromised users In the case of malicious software presence on a system used by the node solutions can be taken into account. Assuming that there is less than X% (exhausting value to estimate at this phase of the project) of malicious software designed for the network. It is indeed difficult to protect people from malicious people. We could add security layers, like a Pin or a passphrase, but if the user is running a keylogger, it will always be insecure. However, since the node can only be accessible from a unique hardware (certificate linked to the hardware), the stolen key could not be as useful as planned. However, the system could also be compromised by someone or something physical like the RUBBER USB[52], in which case the hardware protection could be bypassed. Now we could think of security that only shows a virtual keyboard, but a particular program could sniff the mouse positions and actions. A solution could be to use a USB-key as key, but it could also be replicated if the key is writable. We could use an external hardware, but it would impact Overclouds' public attractivity. Moreover, in the end, how to be sure that the company making the encrypting device will

not be hacked, resulting in the release of the algorithms for generating the keys? Another solution is to control the hardware and the software, assuming that there is no way for an external or internal entity to know what's the key. This last option would mean that Overclouds would have to be privatized somehow, and it would impact nature and vision of the project, by making part of the project proprietary.

Certificate or Identity Clones The network does not allow clones using the network at the same time. The consensus will decide which one is the real one, and the network will not ignore the clone, no peers would accept a connection from it.

In the case an identity is stolen, in the current state of the concept, the attacker would have full access to the identity's assets. However, a solution to avoid this concern would be to link an identity to a unique node. Forcing the user to have a different identity for each node it connects to. The attacker would have to have a clone of the identity key and a clone of the node certificate, which increase the difficulty of the attack. However, it is still possible if the attacker has access to the original hardware and have the technical knowledge to emulate the hardware running the clone of victim's node certificate. To increase the security again and solve this problem, we would have to add an extra layer of protection with a secure additional hardware such as a NitroKey[80]. Now, those solutions are considered extreme, in general, going this far into security is not necessary. However, if those solutions enter into consideration, we could predict a substantial impact on new users willing to join the network.

Note that the consensus could be able to revoke a key or certificate. Protocol for asking the consensus to revoke an identity could be implemented. Alternatively, simply asking many people to blame the stolen identity or certificate.

Answer to unanswered questions from the String Project

- Should we take into account that the wired Internet speed only improves?

  ... No, the speed is not important in our case except maybe during the first load of the Webgate, which is permanently cached into the browser then. The only problem that could happen to a user with a

small bandwidth is that the data will be shared and downloaded at a slower rate. For the other Overclouds users, the slow connections could note even be noticed if they are the seeder is not unique, due to the BitTorrent protocol.

- Should we start from a programmable network such as Ethereum or MaidSafe? Alternatively, should we start from scratch?

  ... It was decided that we would use Ethereum because of its popularity and community. We discovered that help can be found with patience since the project is still under heavy development and attacks on the network are done. Starting from scratch would require a heavy amount of work, which cannot be filled with the time provided for the bachelor project. However, the solutions found with Ethereum at the moment is good enough for the current view of the project.

- Would it be possible to allow nodes to run programs on the network like on Ethereum or with an API like on Maidsafe[71].

  ... That's part of the architecture now.

2.8 Webgate Architecture

Born from the need to create the first link between users, the Webgate is the result of the gateway concept from the global architecture. Its goal is to deliver an easy way to connect to the network right from the browser. The current architecture is found on figure 4 from the annexes, as well as the concept on figure 13.

2.8.1 User

When connecting to the webgate, the user have the choice to either identify himself and access its private data or use the service anonymously.

2.8.2 Background

While connected to the network, a background service runs on the node (browser) which provides the distributed content for the rest of the network. It downloads chunks of encrypted data and distributes them to the network automatically. This service assures that data is this accessible even if the main seeder is not connected anymore.

2.8.3 Webapps

On Overclouds, everything is considered as Webapps. The user is free to either create a webapp or not while uploading content. However, content that is not detected as a webapp will be converted automatically. This concept allows the system to manage the data as folders and have a unique hash address per upload.

2.8.4 Webgate features

- Content
  - Load and Create webapps
  - Encrypt and Decrypt shared content
  - Distribute encryption keys into the network
  - Split and Merge distributed chunks from the webapps
- Background

  - Load, seed and update redundancy table
  - Automatic. It doesn't require the user to do anything
  - Anonymous. The data chunks are encrypted and useless for the user

- Webapps
  - Static storage for media and webapps without database requirements.
  - Dynamic storage for webapps with database requirements. The database is distributed into the network and updated via the Abstract Master Node. The figure 12 from the annexes relates to the dynamic webapps.

3 Implementations

3.1 Consensus

We worked with Ethereum for the consensus. We were able to create and deploy a scalable private blockchain using the clients made in the following languages *Rust, Go, and Cpp* in docker. The tracking of the node is done using the official monitor webapp.

Sadly during the deployment using the Javascript client, a lot of problems occurred, we contacted the developers via the Gitter chat messaging, but they weren't able to us solve the problems such as the bootstrap connection.

Tests We could test the Go, C++ and Rust clients on Mac and Raspberry Pi. Ethereum is not anonymous as it is. The port is 30303, and the IP is not protected.

nodeJS client We tried to set up and test the official ethereumJS; however, it was a pain. No documentation, no tutorials were available. The teacher and the student spent hours on trying to understand and debug, and we are still not sure how it works, the client returns no outputs. The blockchain folder was not increasing either. The developers couldn't help us either.

Docker We could set up a Docker solution for a scalable private blockchain. Ethereum Network Status included. However, we couldn't attach to the Docker cluster from a Mac or a Raspberry Pi.

Browser only We at the NodeJS and Browserified solution.

Javascript APIs We could test the basic Javascript APIs for the Go, Rust, and C++ clients.

3.1.1 Docker Deployement

Execute the commands from the root folder.

Build

- command: docker-compose build

Mount and Run

- command: docker-compose up -d

Scale

- command: docker-compose scale eth=3

Open the netstats

- command: open http://$(docker-machine ip default):3000

Unmount

- docker-compose down

3.2 Webgate

It is the first and the main interface for a user to join and be part the Overclouds network. To join the network, the user must enter the URL `https://overclouds.ch` into the browser bar address and the connection is made. The concept UI can be found in the annexes on the figure 10. It works only with Chrome at the moment, couldn't make it work with Firefox yet.

P2P Research has been done in the Webtorrent direction. We are using the Webtorrent[103] technology, which is the javascript and Webrtc implementation of the bittorrent protocol. Learning and testing the different type of Distributed Hash Tables. Learning and testing bit/web torrent protocol. Research on mobile browsers webrtc capable.

Tracker To simplify the data hash table process, we made Overclouds compatible with the standard webtorrent tracker. This allows the network to be deployed on virtually an unlimited amount of trackers without having to maintain a specific tracker implementation or even have to take care of the servers ourselves.

Offline In the view of having a gateway always accessible, even if the hosting servers are down, the whole website has been made compatible with browser's permanent web storage. The first time the user connects to the webgate, it's ready for offline use. Note that internet is still required to use the service. Also, note that the webgate use Cloudflare for caching the data and for the load balance.

Quick share The user can drag & drop file-s, or folder-s and starting sharing them right from the webgate.

Webapps While sharing content if the "index.html" file is not found, it will automatically create one.

Loading As everything is considered as webapps on the network, each hash address lead the user to some sort of website.

Encryption Currently only the file encryption has been implemented. The user can generate a random key or use its own. After comparing SJCL[95] and CryptoJS[20] libraries, we choose to use CryptoJS because of it was much simpler to use, and the performances were similar even if the on the Table 3 from the annexes showed significant differences.

Web worker The magic allowing decentralized webapps is the web workers fetching the data in parallel.

The webgate always in a tab Due to the background service specification, we force the user always to have the webgate in a tab of the browser. Plus in a technical side, Chrome support webworks only if the main tab is open.

3.2.1 Build and Deployement

As it was developed with NodeJS, the project must be compiled before deploying, once compiled the user may choose to use a localhost server using NodeJS http server or deploy it on a distant server. Execute the commands from the root folder.

Install dependencies (to do once)

- command: npm install

Build

- command: npm run-script build

Localhost

- command: npm run-script start

Server Note that the server and the domain name must be certified with a SSL certificate. We are using Let's encrypt[3], which is free.

- Move the files from the static folder into the root of the distant server.

Patch & Run

- command: npm version patch && npm run-script build && npm run-script start

4 Results

4.1 Consensus

Research as theoretically proven that it's possible to use Ethereum has Overclouds' consensus. However, the implementation right from the browser has not been made, and the contracts are not implemented either.

4.2 Webgate

We could use the WebTorrent[103] protocol in combination with our serverless concept to not relying on signaling servers to connect users together. The service is virtually accessible at any time as long as the user have an internet connection. The data can be shared encrypted and loaded as webapps. However, the decryption implementation has not been made.

4.3 Iterations

Below the results for each weekly iterations (sprints)

Sprint 1 The consensus concept based on Ethereum smart contracts and planning for the next sprints.

Sprint 2 Testing and deploying ethereum with go, rust, and cpp clients. Finding a lot of issues with the Javascript implementation.

Sprint 3 Mesh of trust concept introduced the proof of participation, due to unsolvable problems with ethereum javascript client a replanning has been made, the focus for the next sprint is on the decentralized storage.

Sprint 4 Research on decentralized storage. The best option was to use webtorrent due to the bittorrent protocol and the existing browserified version. In parallel, we are still trying to fix your ethereum problems (DAO attack didn't help at all).

Sprint 5 Made and deployed a basic implementation for decentralized webapp using Webtorrent.

Sprint 6 Data tribunal and data management research lead to the webgate concept.

Sprint 7 Webgate implementation with quick share, SSL encryption, and a webapp demo for videos.

Sprint 8 Finalizing the Webgate and adding encryption for the data sharing.

4.4 Webgate UI

Starting from the concept UI on the figure 10 from the annexes, we are now providing a modern UI for sharing and load content on Overclouds. The final version is of course still a prototype, however, accessible from `https://overclouds.ch`. Screenshots from the prototype UI are found on figures from the annexes.

5 Conclusion

Overclouds' project is fascinating and challenging, there is much material to research and learn. It was said in the specification that the wheel should not be reinvented, it is true, much work and progress has been made in this field for the past six years. As it is until now, it is just a matter of fusing the knowledge to innovate in the field.

Planning The required man hours have been respected. The project has met unplanned problems with Ethereum while working with the javascript implementation. However, the focus has been put on the storage and the results looks promising.

Architecture The architecture evolved from the spring session, and we believe that Overclouds' big picture is respected.

Encryption Javascript is capable of providing client side data encryption required for the transactions and storage.

Storage The bittorrent javascript implementation, webtorrent, has proven to work and responding to our current need.

Consensus The ethereum javascript implementation didn't allow us to provide much work into the consensus itself, however it theoretically possible.

Webgate We implemented a gateway accessible from the browser and allowing to share and load webapps.

6 Bibliography

References

[1] EigenTrust.

[2] Honeypot.

[3] Let's Encrypt.

[4] Penetration Testing Lab.

[5] Prism-Break.

[6] TeamSpeak.

[7] Two-factor authentication.

[8] XBee digimesh.

[9] Filecoin: A Cryptocurrency Operated File Storage Network. 2014.

[10] ArkOS. arkOS.

[11] Ayeowch. GLOBAL BITCOIN NODES DISTRIBUTION.

[12] F. Bellard. Javascript PC Emulator, 2015.

[13] J. Benet. IPFS-Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*, (Draft 3), 2014.

[14] J. Benet and J. Ai. IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3).

[15] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake. 42(240258):1–19, 2013.

[16] P. Biddle, P. England, M. Peinado, and B. Willman. The Darknet and the Future of Content Protection. *Digital Rights Management Technological, Economic, Legal and Political Aspects*, pages 344–365, 2003.

[17] Bitmessage. Bitmessage.

[18] BitTorrent. Project Maelstrom.

[19] Boum. Privacy for anyone anywhere.

[20] Brix. CryptoJS, 2013.

[21] V. Buterin. Slasher Ghost, and Other Developments in Proof of Stake, 2014.

[22] V. Buterin. Understanding Serenity, Part 2: Casper, 2015.

[23] Cjdns. cjdns.

[24] R. Claret. 16DLM-TP210 Overclouds, 2016.

[25] I. Clarke and Et Al. A distributed decentralised information storage and retrieval system. *Undergraduate Thesis*, 1999.

[26] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Designing Privacy Enhancing ...*, 23:46–66, 2001.

[27] COLIN PERCIVAL. STRONGER KEY DERIVATION VIA SEQUENTIAL MEMORY-HARD FUNCTIONS. 2012.

[28] Commotion. Community Wireless Networks.

[29] Cuonic. PeerShare.

[30] L. Delmer. *L'emergence au sein d'internet de communautes virtuelles et anonymes, Freenet et i2p*. PhD thesis, Universite catholique de Louvain - Departement des sciences politiques et sociales, 2009.

[31] Diaspora. Diaspora*.

[32] I. Digital Bazaar. forge, 2016.

[33] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. *SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium*, 13:21, 2004.

[34] DR. GAVIN WOOD. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER HOMESTEAD DRAFT. 2015.

[35] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. *Advances in Cryptology—CRYPTO'92*, pages 139–147, 1993.

[36] S. Edward. Snowden Digital Surveillance Archive.

[37] Ethereum. Ethereum Homestead Documentation, 2016.

[38] Ethersphere. IPFS SWARM.

[39] A. H. Ferreira, R. L. Pereira, and F. M. Silva. Content redundancy in BitTorrent. In *2012 21st International Conference on Computer Communications and Networks, ICCCN 2012 - Proceedings*, 2012.

[40] FlowingMail. FlowingMail.

[41] R. Foodists. The Underground Internet. 2003.

[42] Freedom.js. freedom.js, 2012.

[43] Freenet. Freenet project.

[44] G. Lawrence Paul Sundararaj1 D. R. Anita Sofia Liz2. Anti-Sybil Mechanism against Bogus Identities\nin Social Networks. *International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)*, 01(02):123–127, 2014.

[45] M. Gabel, C. Brigham, A. Cheyer, and J. Levy. Dynamically evolving cognitive architecture system based on third-party developers. *United States Patent Application*, 1(20150100943), 2008.

[46] O. Garden. Open Garden.

[47] P. Gardner-stephen. The Serval Project : Practical Wireless Ad-Hoc Mobile Telecommunications. (June):1–29, 2011.

[48] Google. Web RTC.

[49] Google. Closure Library, 2015.

[50] J. Guo, P. Karman, I. Nikolić, L. Wang, and S. Wu. Analysis of BLAKE2. *Springer International Publishing Switzerland 2014*, 8366(8366):402–423, 2014.

[51] HacDC. Byzantium.

[52] Hakshop. Rubber Ducky USB.

[53] M. Hayward. PeerSurf.

[54] Hookflash. Open Peer, 2014.

[55] Hubzilla. Hubzilla.

[56] Hype. Hyperboria Whitepaper.

[57] Hyperboria. Hyperboria.

[58] I2P. The Invisible Internet Project.

[59] IIP. Invisible IRC Project, 2003.

[60] I. Info Tech. Digital Signature in the Browser, 2014.

[61] I. Info Tech. Symmetric Encryption Sample, 2014.

[62] IPFS. The IPFS Project.

[63] M. Jakobsson and A. Juels. Proofs of work and bread pudding protocols (extended abstract). *Secure Information Networks*, pages 258–272, 1999.

[64] P. Johnston. jsHashes, 2015.

[65] S. King and S. Nadal. Peercoin, 2012.

[66] S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. *Ppcoin.Org*, 2012.

[67] W. Koch. Using the GNU Privacy Guard. *Notes*, (March), 2008.

[68] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[69] Litecoin. Litecoin Wiki, 2011.

[70] MaidSafe. MaidSafe.

[71] MaidSafe. MaidSafe.net announces project SAFE to the community, 2014.

[72] Max K., Patrick Lodder, and Ross Nicoll. Dogecoin Core, 2013.

[73] MDN. MDN Web API Crypto, 2015.

[74] MediaGoblin. MediaGoblin.

[75] N. Mehta. The Heartbleed Bug, 2014.

[76] Microsoft. MSR JavaScript Cryptography Library, 2015.

[77] J. Minardi. SyncNet: A Decentralized Web Browser.

[78] Movim. Movim.

[79] L. Nick. CONSENSUS WITHOUT A BLOCKCHAIN, 2015.

[80] Nitrokey. Nitokey.

[81] OpenWebRTC. OpenWebRTC.

[82] Peer5. Peer5, 2015.

[83] PGP. PGP ® White Paper PGP ® Global Directory. (August), 2005.

[84] C. Point and S. Technologies. VPN Administration Guide R76. (August), 2014.

[85] E. R. Stewart. Stream Control Transmission Protocol. *Network Working Group*, 2007.

[86] Retroshare. Retroshare.

[87] E. Saarinen, M-J. and J. P. Aumasson. The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC), 2015.

[88] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

[89] Serval. Serval Project.

[90] P. Server. Peer Server, 2013.

[91] Sinch. ORTC VS WEBRTC — WHAT'S THE DIFFERENCE, 2015.

[92] G. Sivek, S. Sivek, J. Wolfe, and M. Zhivich. WebTorrent: a BitTorrent Extension for High Availability Servers. 2004.

[93] R. Sleevi and M. Watson. Web Cryptography API, 2014.

[94] Y. Sompolinsky and a. Zohar. Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. *Eprint.Iacr.Org*, pages 1–31, 2014.

[95] E. Stark, M. Hamburg, and D. Boneh. Symmetric cryptography in javascript. In *Proceedings - Annual Computer Security Applications Conference, ACSAC*, pages 373–381, 2009.

[96] N. Sullivan. Staying ahead of OpenSSL vulnerabilities, 2014.

[97] M. Switch. PeerJS.

[98] D. Tarr. Crypto-Browserify, 2013.

[99] D. Tarr. Performance of Hashing in Javascript Crypto Libraries., 2014.

[100] Tent.is. Tent.io, 2013.

[101] TheBitcoinNews. Bitcoin Spam Attacks, 2015.

[102] Tor. Tor.

[103] W. Torrent. Web Torrent, 2015.

[104] S. University. Stanford Javascript Crypto Library, 2009.

[105] Vitalik Buterin. A Next-Generation Smart Contract and Decentralized Application Platform. 2013.

[106] J. Warren. Bitmessage: A Peer-to-Peer Message Authentication and Delivery System. page 5, 2012.

[107] C. Webber. GNU MediaGoblin Documentation. 2016.

[108] D. Wei. B-Money, 1998.

[109] S. Wilkinson. Storj A Peer-to-Peer Cloud Storage Network.

[110] YaCy. Web Search by the people, for the people.

[111] ZeroNet. ZeroNet.

[112] Zeronet. ZeroNet, 2016.

7 Annexes

7.1 State of the Art

For the project initiation, it was important to do research with the goal of targeting the needs for existing knowledge and technologies. That information could potentially be used to help achieve this project. This subsection will expose the research done.

7.1.1 Similar products

To be straight forward. A comparable project working right from the browser without the help of any third party or background software is nonexistent. At least not from the public knowledge available with our research keywords.

7.1.2 Existing Networks

The most common form of networks approaching our project's vision are called *Darknets*[16] and they started to emerge during the years 2000ish[30]. They are all aiming to encrypt data transmissions and protect network's users from being spied on and bypass censorship.

Freenet[25, 26] Description from the official website [43]

> Freenet is free software which lets you anonymously share files, browse and publish *freesites* (web sites accessible only through Freenet) and chat on forums, without fear of censorship. Freenet is decentralised to make it less vulnerable to attack, and if used in *darknet* mode, where users only connect to trusted nodes (real life friends, etc.), is very difficult to detect.
>
> Communications on Freenet nodes are encrypted and are routed through other nodes to make it extremely difficult to determine who is requesting the information and what its content is.
>
> Each user contributes to the network by giving bandwidth and a portion of their hard drive for storing files. Files are encrypted, so generally the user cannot quickly discover what is in his *datastore*. Chat forums, websites, and search functionality, are all built on top of this distributed data store.

I2P[41] Description from the official website [58] Originally IIP[59].

> The Invisible Internet Project is an anonymous network, exposing a simple layer that applications can use to anonymously and securely send messages to each other. The network itself is strictly message based, but there is a library available to allow reliable streaming communication on top of it (a la TCP). All communication is end to end encrypted (in total there are four layers of encryption used when sending a message), and even the end points (*destinations*) are cryptographic identifiers (essentially a pair of public keys).

MaidSafe[71] Description from the official website [70]

> The SAFE (Secure Access For Everyone) Network is made up of the unused hard drive space, processing power and data connection of its users. It offers a level of security and privacy not currently available on the existing Internet and turns the tables on companies, putting users in control of their data, rather than trusting it to organizations.

Tor[33] Description from the official website [102] "Tor is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security."

ZeroNet[112] Description from the official website [111]

> Real-time updated, P2P websites using Bitcoin cryptography and the BitTorrent network. Zeronet is decentralized, open source software in Python aimed to build an Internet-like computer network of peer-to-peer users. It is not anonymous by default, but users can hide their IP address by using Tor which uses Bitcoin cryptography and the BitTorrent network.

Project Maelstrom Description from the official website [18]

> BitTorrent wants your help creating a P2P-powered web with Project Maelstrom. (Beta on Windows only) The Maelstrom Network - currently under hectic development - is gearing up to be able to provide users and developers a joined interface to each other. Unlike some distinct offerings out there, this isn't a social network, and it never will be. All your information is for your eyes only until you allow an application to use it. We're trying to get web applications to a first class status on the web, similar to how an application on your computer is tightly integrated into the rest of the computer.

Diaspora* Description from the official website [31] " The community-run, distributed social network. diaspora* is based on three key philosophies: Decentralization, Freedom, and Privacy. "

FlowingMail Description from the official website [40]

> FlowingMail is the name of a new decentralized, secure and encrypted email protocol. The most used email systems rely on a central server that receives, stores and forward the messages: FlowingMail is decentralized and does not rely on a central server to deliver the encrypted emails. The scope of the FlowingMail protocol is to hide the information being transmitted and the parties involved in the communication. The main component of the FlowingMail protocol is a Kademlia Distributed Hash Table (DHT), which is responsible for storing the encrypted emails while they are in transit and the certificates of the participants in the FlowingMail network.

Bitmessage[106] Description from the official website [17] "Bitmessage is a P2P communications protocol used to send encrypted messages to another person or to many subscribers. It is decentralized and trustless, meaning that you need-not inherently trust any entities like root certificate authorities."

Retroshare Description from the official website [86]

> RetroShare is free software for encrypted filesharing, serverless email, instant messaging, chatrooms, and BBS, based on a friend-to-friend network built on GPG. It is not strictly a darknet since optionally, peers may communicate certificates and IP addresses from and to their friends.

MediaGoblin[107] Description from the official website [74] "MediaGoblin is a free software media publishing platform that anyone can run. You can think of it as a decentralized alternative to Flickr, YouTube, SoundCloud, etc."

Movim Description from the official website [78]

> My Open Virtual Identity Manager is a distributed social network built on top of XMPP, a popular open standards communication protocol. Movim is a free and open source software licensed under the AGPL. It can be accessed using existing XMPP clients and Jabber accounts.

The IPFS Project[13] Description from the official website [62]

> The InterPlanetary File System is a new hypermedia distribution protocol, addressed by content and identities. IPFS enables the creation of completely distributed applications. It aims to make the web faster, safer, and more open. IPFS claims to be a Permanent Web with a new peer-to-peer hypermedia protocol.

Serval Project[47] Description from the official website [89] "Serval is a telecommunications system comprised of at least two mobile phones that are able to work outside of regular mobile phone tower range due thanks to the Serval App and Serval Mesh."

Open Garden Description from the official website [46]

> Open Garden's technology creates a software-based network, also known as a peer-to-peer wireless mesh network, among participating mobile devices using WiFi, Bluetooth LE, and other technologies. Open Garden's innovations include: seamless device discovery and pairing, offline identity, a proprietary network protocol for addressing and routing messages off-the-grid, distributed algorithms for managing mesh networks, advanced traffic management (multi-hop, store and forward), and battery use reduction.

Hyperboria[56] Description from the official website [57] "A community of local Wifi initiatives, programmers, and enthusiasts. They are running a peer-to-peer IPv6 network with automatic end-to-end encryption, distributed IP address allocation, and DHT-based Source Routing."

7.1.3 Transfer Protocols

Route-based VPN [84] "The use of VPN Tunnel Interfaces (VTI) introduces a new method of configuring VPNs called Route Based VPN. This method is based on the notion that setting up a VTI between peer Security Gateways is much like connecting them directly."

Commotion Description from the official website [28] "Commotion is an open-source communication tool that uses wireless devices to create decentralized mesh networks."

Hubzilla Description from the official website [55] "Hubzilla is a powerful platform for creating interconnected websites featuring a decentralized identity, communications, and permissions framework built using common webserver technology."

Tent Description from the official website [100] "Tent lets you control your data instead of handing it over to service and app providers. Just like email, you choose a provider or can set up your own Tent server. Tent data and relationships are portable so you can change Tent providers easily at any time."

cjdns Description from the official website [23]

> Networking Reinvented. Cjdns implements an encrypted IPv6 network using public-key cryptography for address allocation and a distributed hash table for routing. This provides near-zero-configuration networking and prevents many of the security and scalability issues that plague existing networks.

WebRTC Description from the official website [48] "WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. The WebRTC components have been optimized to best serve this purpose." http://iswebrtcreadyyet.com

Open Peer Description from the official website [54]

> Real-time Communications Protocol & Specification. What further reduces the proliferation of peer-to-peer is a lack of standardization, openness and ubiquity of the technology. The standards bodies have been working for years on firewall traversal techniques and standardization of the approaches and a new joint effort called WebRTC between the W3C and IETF on how browsers can directly communication between browsers to move media. This joint effort does not specify how signalling happens between peers so it's not a complete solutions on its own.

freedom.js Description from the official website [42]

> It's a framework for building peer-to-peer (P2P) web apps. Easily create social applications that work in modern web browsers, Chrome packaged apps, Firefox extensions, node.js, and native mobile apps. freedom.js apps are just JavaScript, so they can be distributed as packages on an app store or hosted on static web servers.

7.1.4 Protections

Honeypot Description from the official website [2]

> In computer terminology, a honeypot is a computer security mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems. Generally, a honeypot consists of data (for example, in a network site) that appears to be a legitimate part of the site but is actually isolated and monitored, and that seems to contain information or a resource of value to attackers, which are then blocked.

Penetration Testing Lab Description from the official website [4] "Use of multiples OS to target penetration type on a network or computer."

Two-factor authentication Description from the official website [7] "That provides identification of users by means of the combination of two different components."

Let's Encrypt Description from the official website [3] "A new Certificate Authority: It's free, automated, and open."

7.1.5 Cryptography

Stanford Javascript Crypto Library[95] [104] "The Stanford Javascript Crypto Library (hosted here on GitHub) is a project by the Stanford Computer Security Lab to build a secure, powerful, fast, small, easy-to-use, cross-browser library for cryptography in Javascript."

MSR JavaScript Cryptography Library Description from the official website [76] "The Microsoft Research JavaScript Cryptography Library has been developed for use with cloud services in an HTML5 compliant and forward-looking manner."

MDN Description from the official website [73]

> The Crypto interface represents basic cryptography features available in the current context. It allows access to a cryptographically strong random number generator and to cryptographic primitives.
>
> An object with this interface is available on Web context via the Window.crypto property.

W3C Description from the official website [93]

> This specification describes a JavaScript API for performing basic cryptographic operations in web applications, such as hashing, signature generation and verification, and encryption and decryption. Additionally, it describes an API for applications to generate and/or manage the keying material necessary to perform these operations. Uses for this API range from user or service authentication, document or code signing, and the confidentiality and integrity of communications.

Google Closure Description from the official website [49]

> The Closure Library is a broad, well-tested, modular, and cross-browser JavaScript library. You can pull just what you need from a large set of reusable UI widgets and controls, and from lower-level utilities for DOM manipulation, server communication, animation, data structures, unit testing, rich-text editing, and more.

forge Description from the official website [32] "The Forge software is a fully native implementation of the TLS protocol in JavaScript as well as a set of tools for developing Web Apps that utilize many network resources."

jsHashes Description from the official website [64] "Fast and dependency-free cryptographic hashing library for node.js and browsers (supports MD5, SHA1, SHA256, SHA512, RIPEMD, HMAC)"

crypto-broserify Description from the official website [98] "The goal of this module is to reimplement node's crypto module, in pure javascript so that it can run in the browser."

BLAKE2[87] Description from the paper Analysis of BLAKE2 [50] "Recently proposed and already in use tweaked version of the SHA-3 finalist BLAKE."

7.1.6 Hardware

XBee digimesh Description from the official website [8]

> XBee & XBee-PRO DigiMesh 2.4 embedded RF modules utilize the peer-to-peer DigiMesh protocol in 2.4 GHz for global deployments. This innovative mesh protocol offers users added network stability through self-healing, self-discovery, and dense network operation. With support for sleeping routers, DigiMesh is ideal for power sensitive applications relying on batteries or power harvesting technology for power.

Nitrokey Description from the official website [80] "USB key to enable secure encryption and signing of data. The secret keys are always stored inside the Nitrokey."

7.1.7 Blockchains

So much hype in this, but what is it? Everybody relate it mainly to *Bitcoin*[88], indeed *Bitcoin* introduced this technology (which is its main innovation), but *Bitcoin* is not equivalent to chain-block.

Indeed, the main idea is that no one controls or owns the chain of blocks and forges a system for electronic transactions without relying on trust. A block contains a timestamp and information linking it to a previous block.

Note that a block looks like digital objects that record and confirm when and in what sequence transactions enter in the block chain.

Blocks created by network users with specialized software or specially designed equipment. Block creator are known as "miners" to reference the gold mining. Bitcoin showed us a pretty amazing evolution in the mining procedure; it was designed at the start by Satoshi Nakamoto for CPUs, then it quickly involved into GPU mining then into programmable chips (FPGA) then now it goes into burned circuits (ASIC).

In a crypto-currency system, miners are incentivized to create blocks to collect two types of rewards: a pre-defined per-block award and fees offered within the transactions themselves, payable to any miner who successfully confirms the transaction. Every node in a decentralized system has a copy of the blocks chain; it avoids the need for a trusted authority to timestamp transactions. Decentralized block chains use various timestamping schemes, such as proof-of-work or more recently with PeerCoin the proof-of-participation.

Bitcoin [88] Why did everybody already hear the word *Bitcoin*? It is the first successful implementation of a distributed crypto-currency as described partially by Wei Dai in 1998[108]. The foundation of Bitcoin is the assumption that money could be anything (object, record, stake, etc.) accepted as payment for goods or services by a consensus (country, social, economical, etc.). Designed with the idea of using cryptography as proof of existence and transfer of virtual assets (proved to exist). Rather than relying on a central authority (trusted third party), Bitcoin is decentralized, meaning that it works with the network consensus. It uses the peer-to-peer technology to operate with the transaction management and verifying that the virtual assets is carried out collectively by the network.

Ethereum [37] Seen today has the new way to use the blockchains technology. It uses the currency has fuel to execute turing-complete smart contracts. Contracts, see as autonomous agents, are programs that run on the Ethereum Virtual Machine. The EVM is being part of the protocol and runs on each client contributing to the network; they are all doing and storing the same calculations. Note that it's not efficient to compute in parallel redundantly, but it offers a consensus for the computed results.

Others We can find a lot of forked crypto-currencies from Bitcoin; everybody is trying to make the success theirs. However, no major changes have been made to them expect the genesis block (the first block that determines how long will be the chain, etc.). We will just cite some of them that have some special modifications. Note also that they don't provide whitepapers.

Lite Coin[69] The major differences with Bitcoin are the time process focus to generate new blocks of 2.5 mins vs. 10 mins for Bitcoin, the use of the Scrypt[27] library and the maximum cap of coins is 84 million (4 times more than Bitcoin).

Dodge Coin [72] It started as a "Joke Currency" but it got capitalized... Its particularity is to have no limits in coins produced, however, the per block reward decreases.

Peer Coin [65] Based on the paper of Scott Nadal and Sunny King [66] for the Proof-of-Stake Peer Coin was born.

7.1.8 Decentralized applications

YaCy Description from the official website [110] "YaCy is a free distributed search engine, built on principles of peer-to-peer networks. Its core is a computer program written in Java distributed on several hundred computers, so-called YaCy-peers."

7.1.9 Reputation Management

TeamSpeak Description from the official website [6]

> TeamSpeak is proprietary voice-over-Internet Protocol software that allows computer users to speak on a chat channel with fellow computer users, much like a telephone conference call. (Allow the user to create a mathematically generated identity certificate, with a level increasing over time)

EigenTrust Description from the official website [1]

> EigenTrust algorithm is a reputation management algorithm for peer-to-peer networks. The algorithm provides each peer in the network a unique global trust value based on the peer's history of uploads and thus aims to reduce the number of inauthentic files in a P2P network. It has been cited by approximately 3853 other articles according to Google Scholar.

7.1.10 Operating Systems

Project-Byzantium Description from the official website [51] "Byzantium is a live Linux distribution that delivers easy-to-use, secure, and robust mesh networking capabilities."

Tails Description from the official website [19] "Tails is a live operating system, that you can start on almost any computer from a DVD, USB stick, or SD card. It aims at preserving your privacy and anonymity."

arkOS Description from the official website [10] "Your personal server sets sail. Take control of your online life by easily hosting your own server apps: websites, email, files and much more. Featuring one-click installs and rolling updates. Store at home or with a decentralized provider."

7.1.11 Technologies

QEMU JS [12] "It is a generic and open source machine emulator and virtualizer."

PeerJS Description from the official website [97]

> PeerJS is a service which makes it easier to build a chat room using the present WebRTC's PeerConnection API. The PeerConnection API proposes to be able to send data, video etc from one user-agent to another without the need for it going through a server. PeerJS handles this handshake with a simple Socket.IO backend server.

PeerShare Description from the official website [29] "PeerShare is a P2P file sharing website that uses WebRTC technologies to allow uses to send and receive files without going through any servers. PeerShare is mainly built on Javascript and jQuery and uses PeerJS as a WebRTC API."

PeerSurf Description from the official website [53] "PeerSurf is a demo (and kind of a library) of P2P websites powered by WebTorrent"

OpenWebRTC Description from the official website [81]

> With OpenWebRTC you can build native WebRTC apps that communicate with browsers that supports the WebRTC standard, such as Chrome, Firefox and Bowser. OpenWebRTC is especially focused on mobile platforms, with powerful features such as hardware accelerated video coding and OpenGL-based video rendering.

PeerServer Description from the official website [90]

> This system allows you to quickly create a decentralized, short-lived web application where all the content lives within your browser. The traditional server only performs the initial handshake between the client-browsers and the client-server; your browser serves all other content peer-to-peer.

7.1.12 Other

Prism-Break Description from the official website [5] "Help make mass surveillance of entire populations uneconomical! We all have a right to privacy, which you can exercise today by encrypting your communications and ending your reliance on proprietary services."

7.2 Architectures

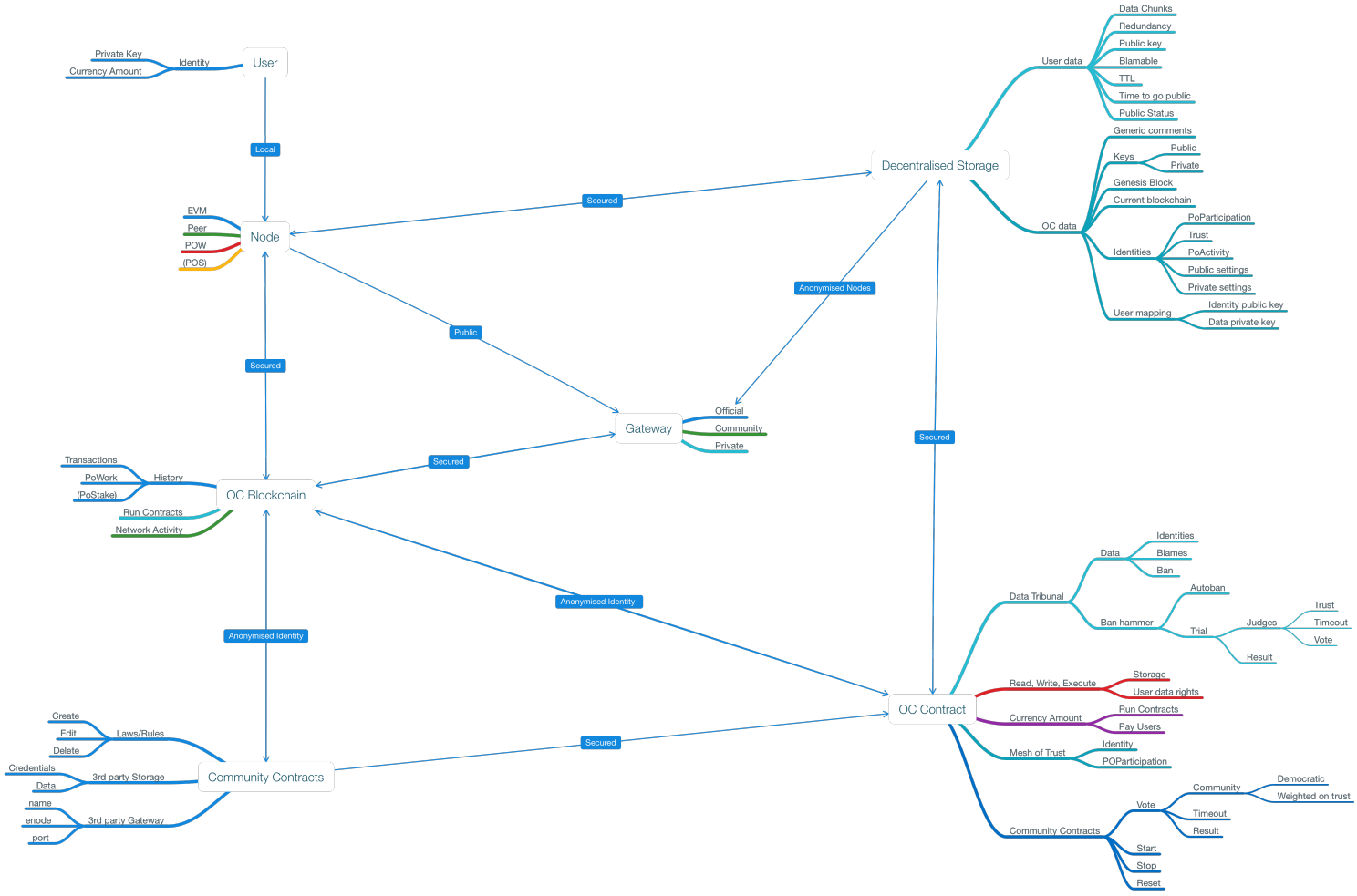


Figure 1: Overclouds latest global architecture

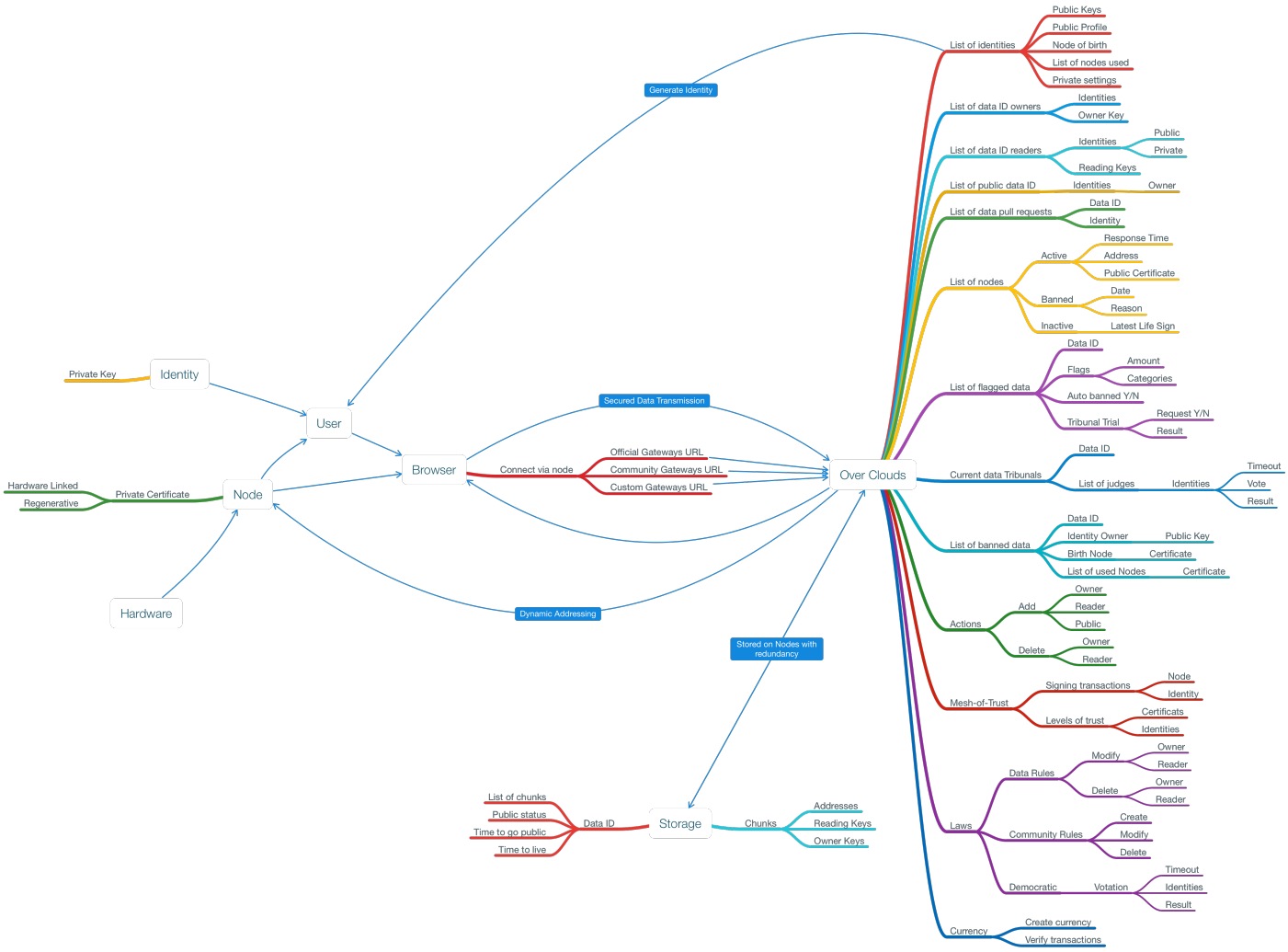


Figure 2: Overclouds latest global architecture from spring session

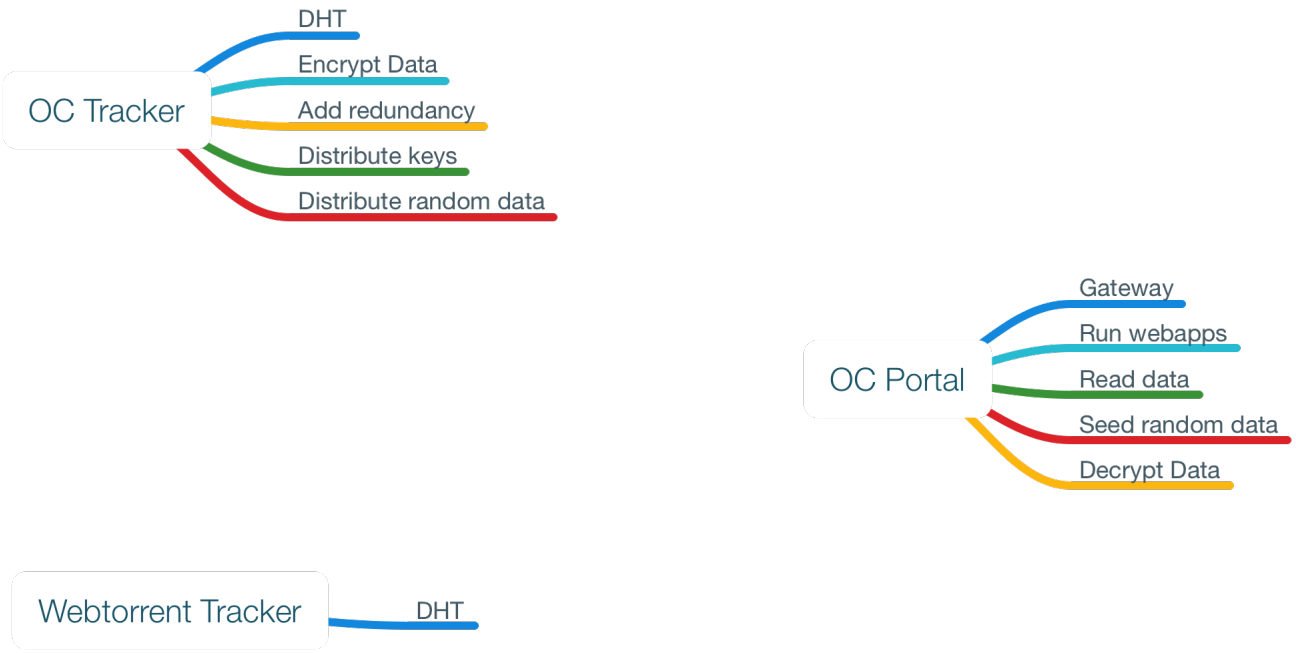


Figure 3: Storage architecture

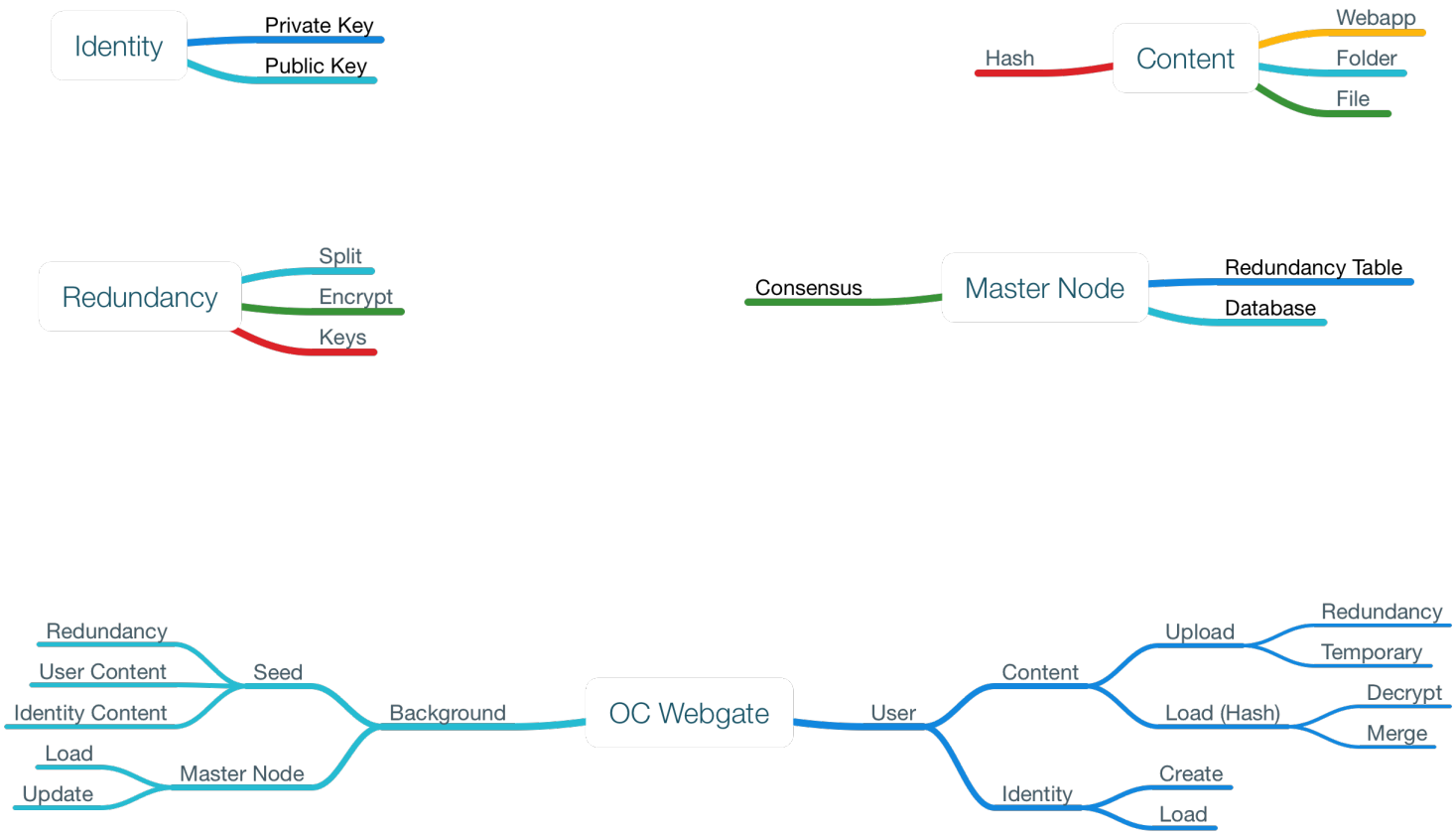


Figure 4: Webgate architecture

7.3 Concepts

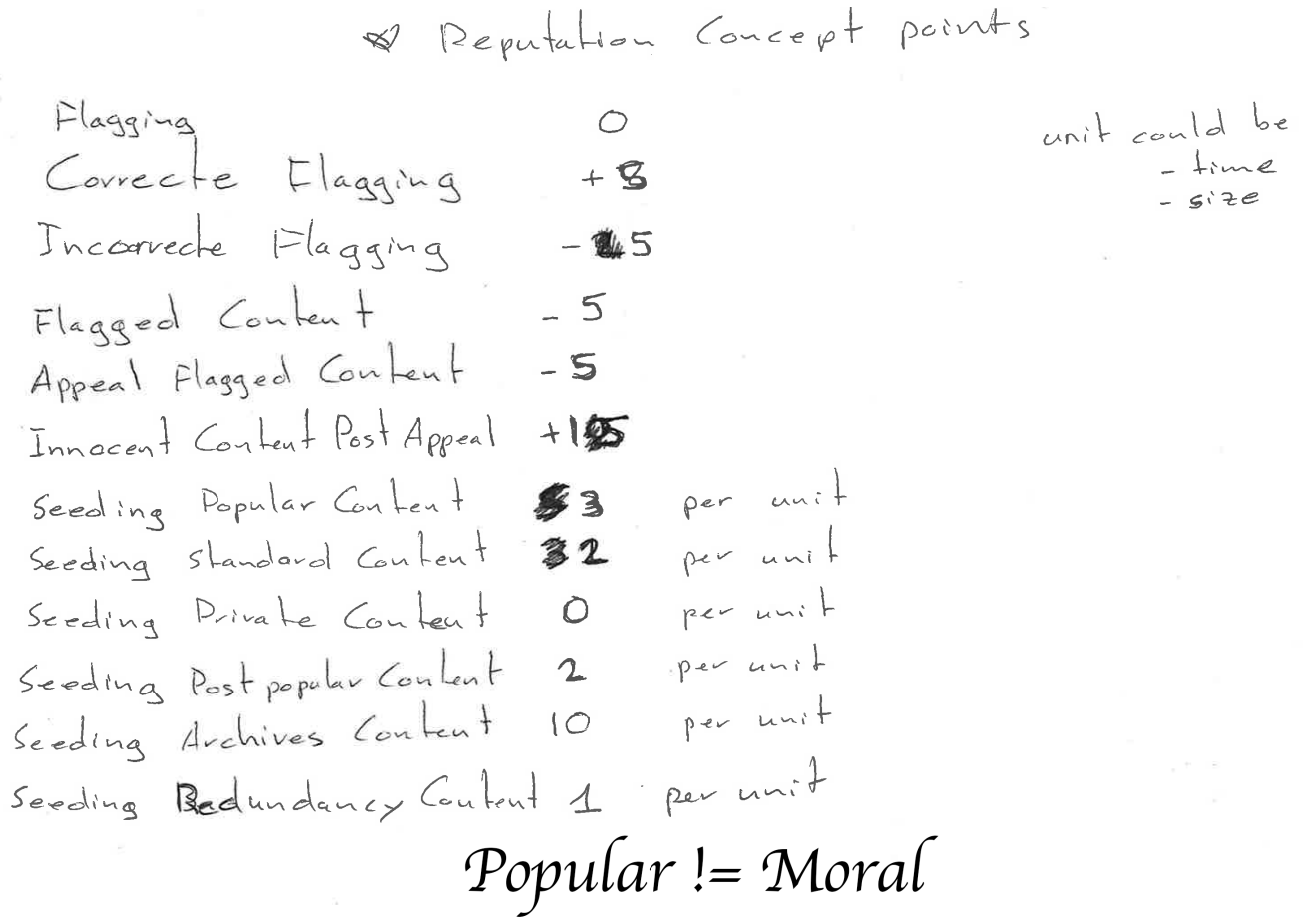


Figure 5: Concept for data reputation points

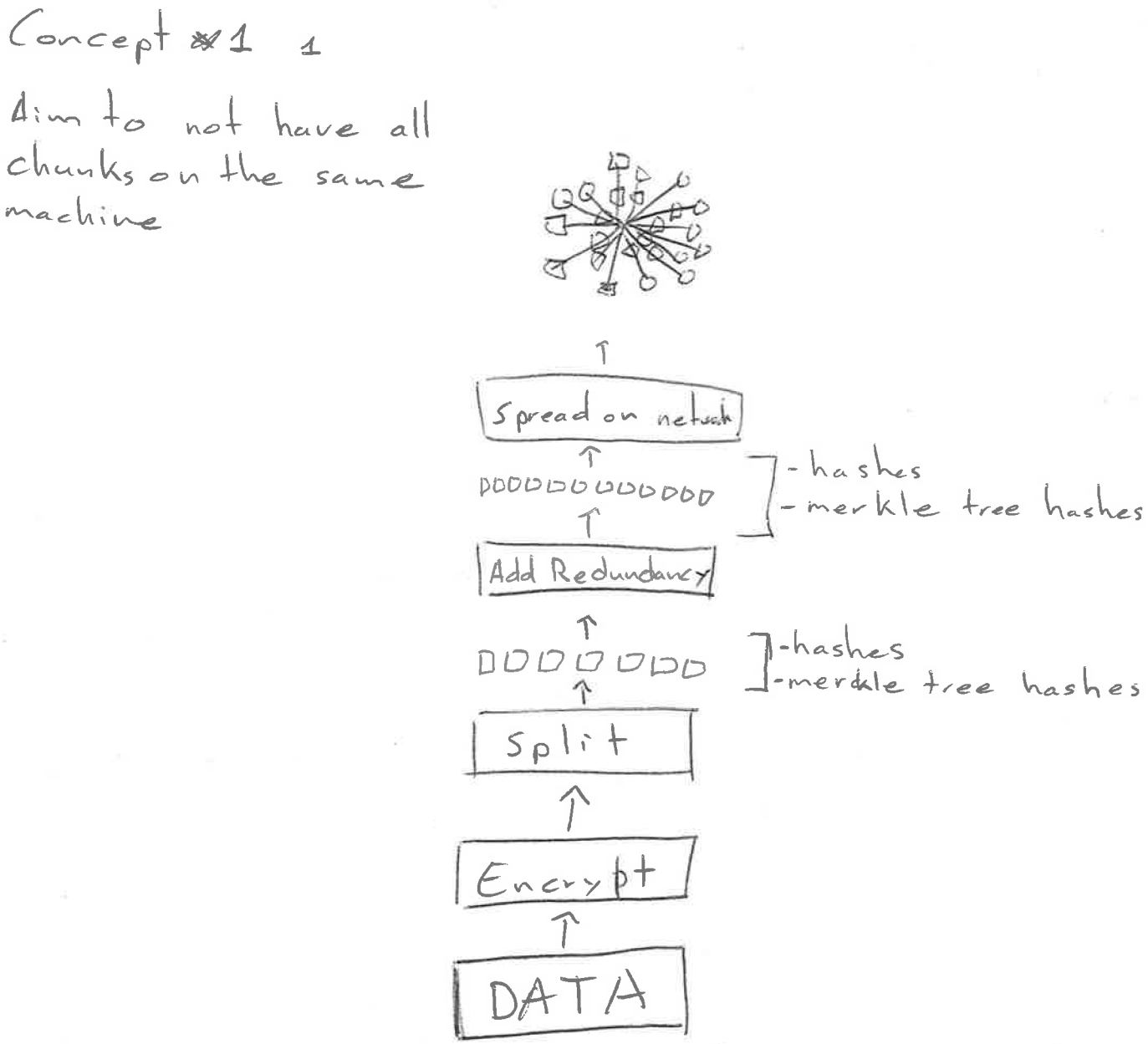


Figure 6: Concept for storage 1

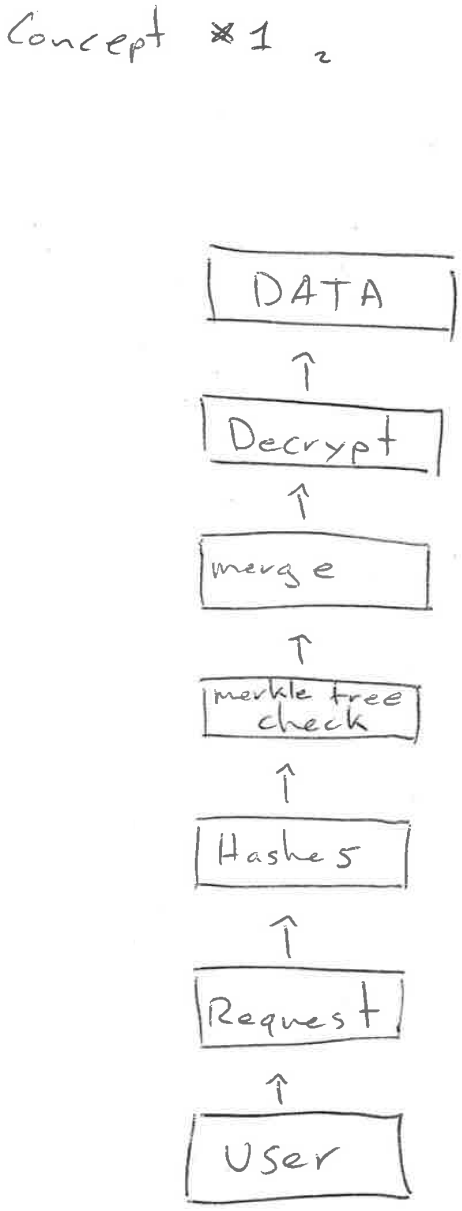


Figure 7: Concept for storage 2

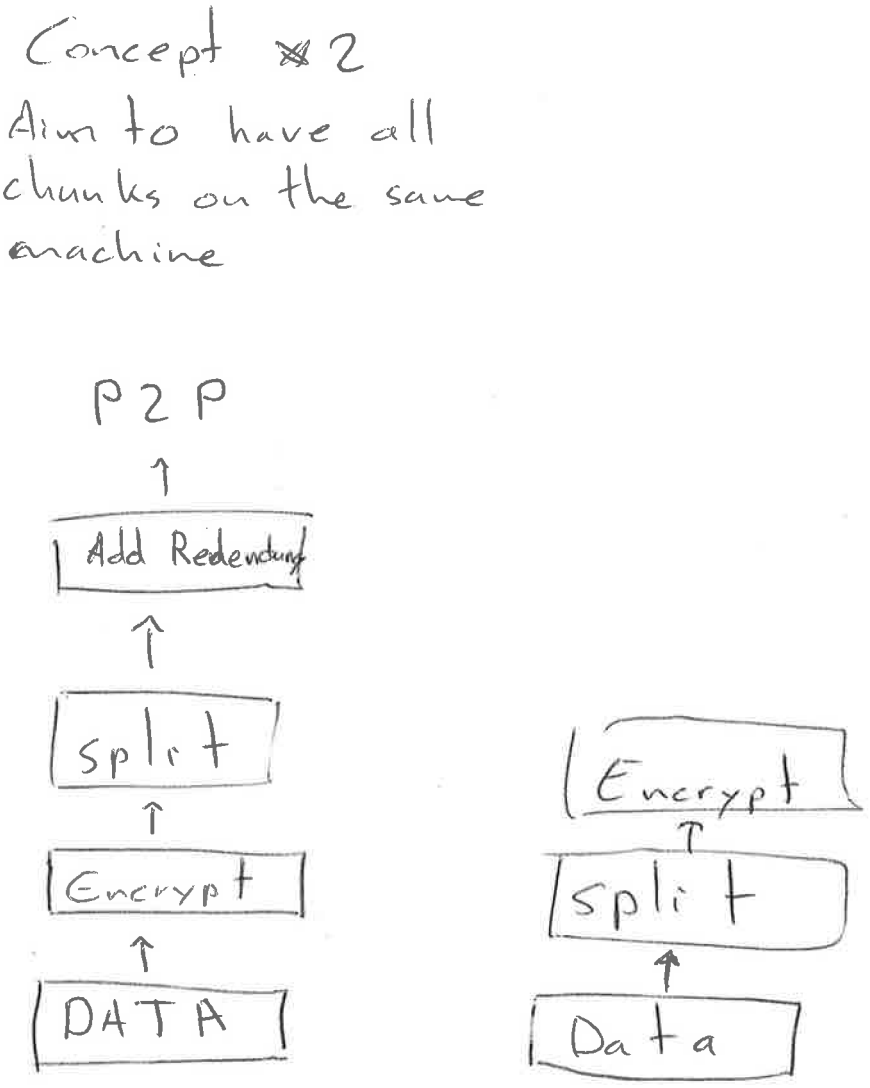


Figure 8: Concept for storage 3

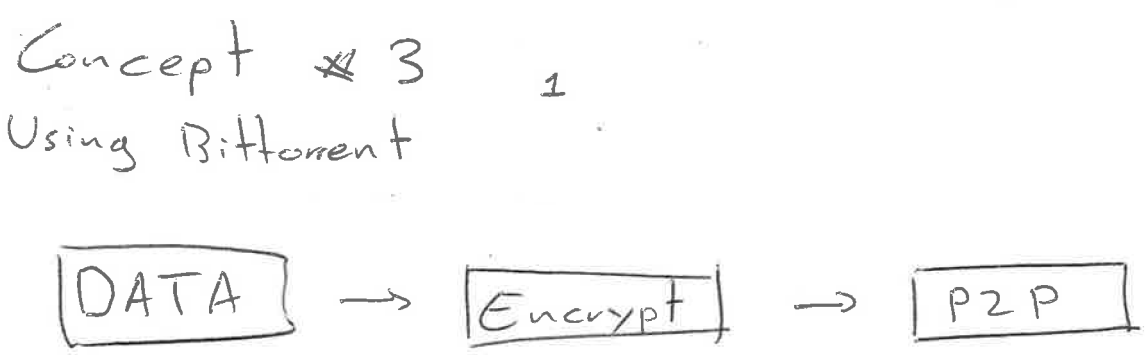


Figure 9: Concept for storage 4

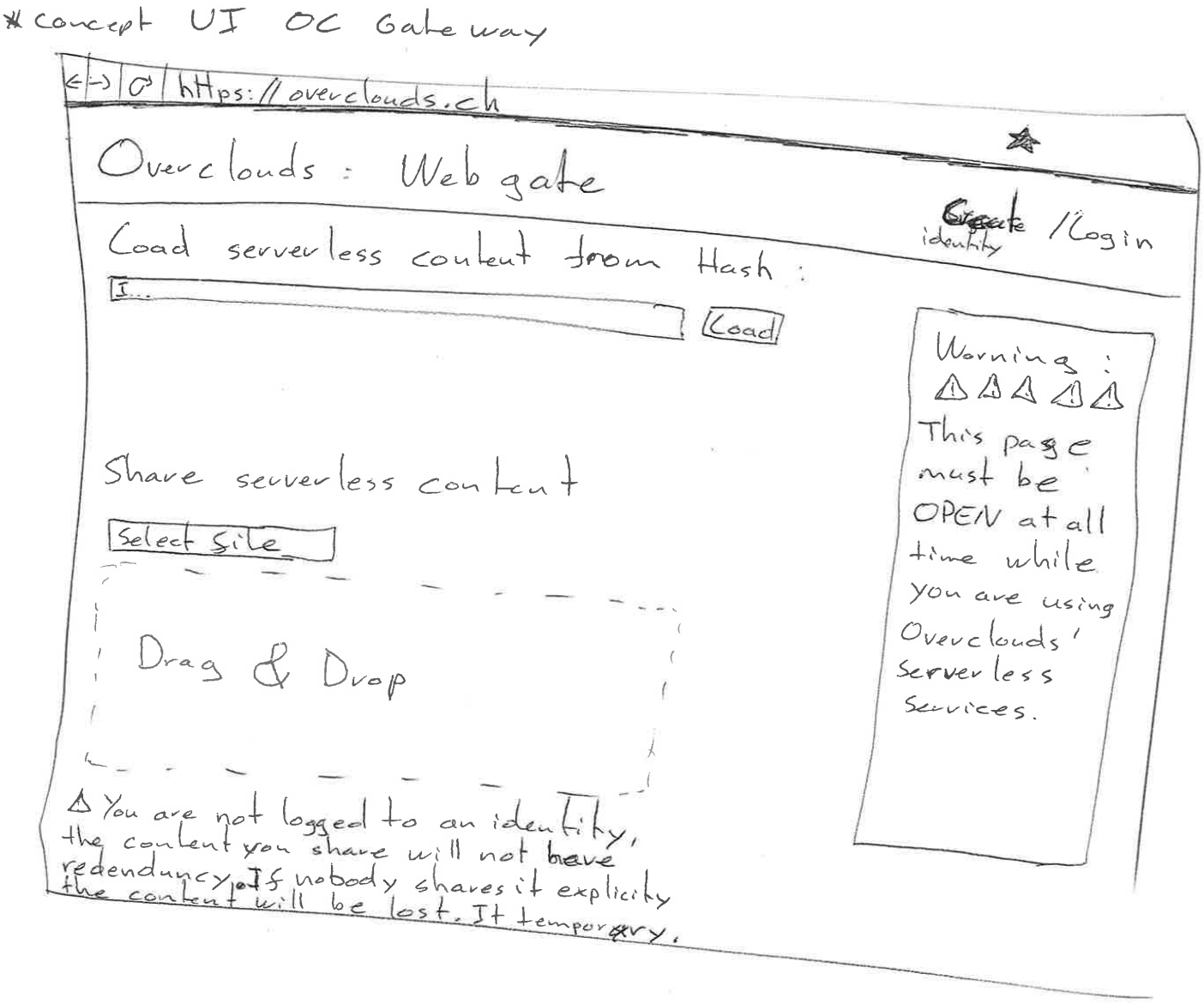


Figure 10: Concept of the webgate UI

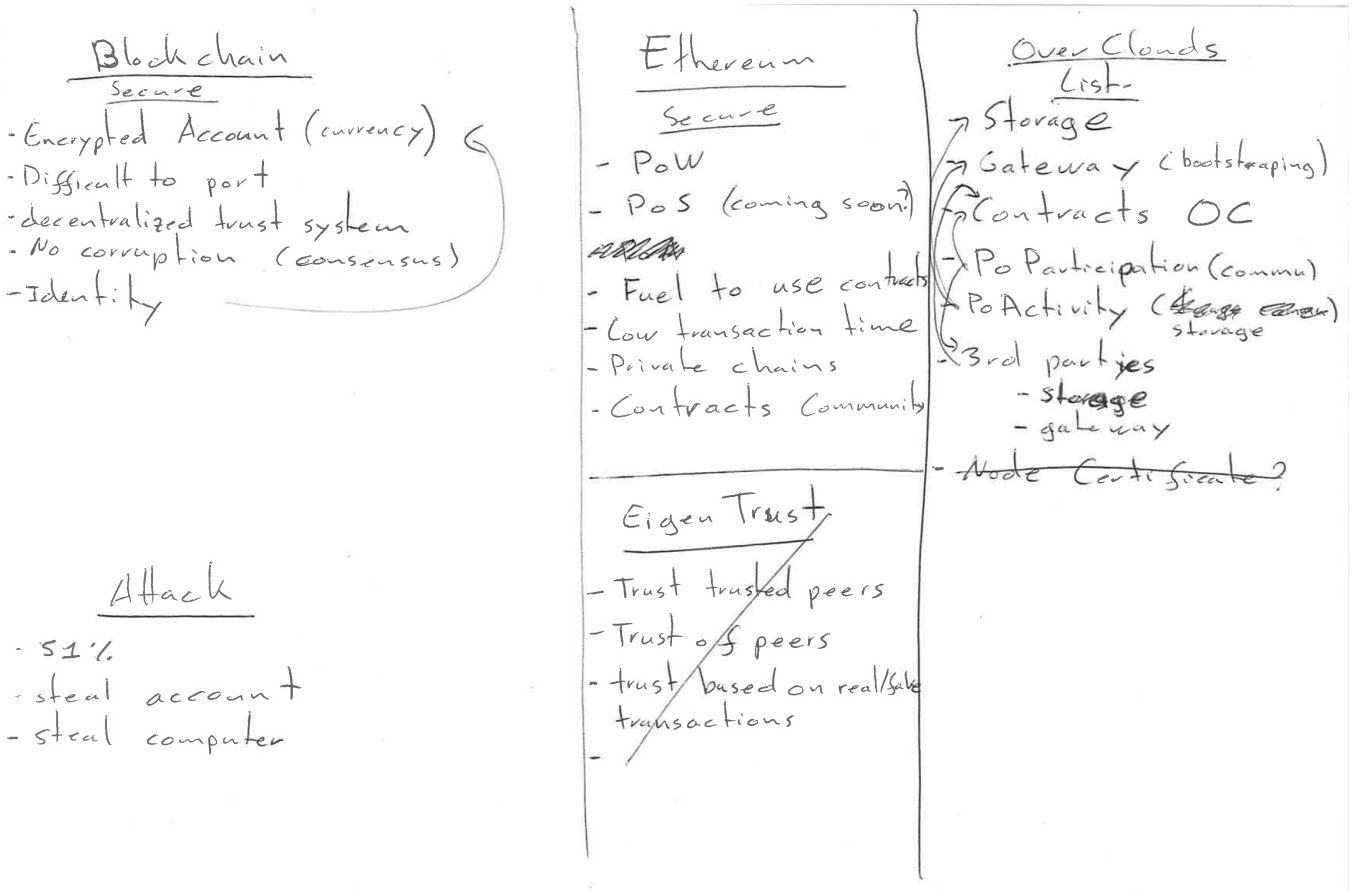


Figure 11: Brainstorming trust

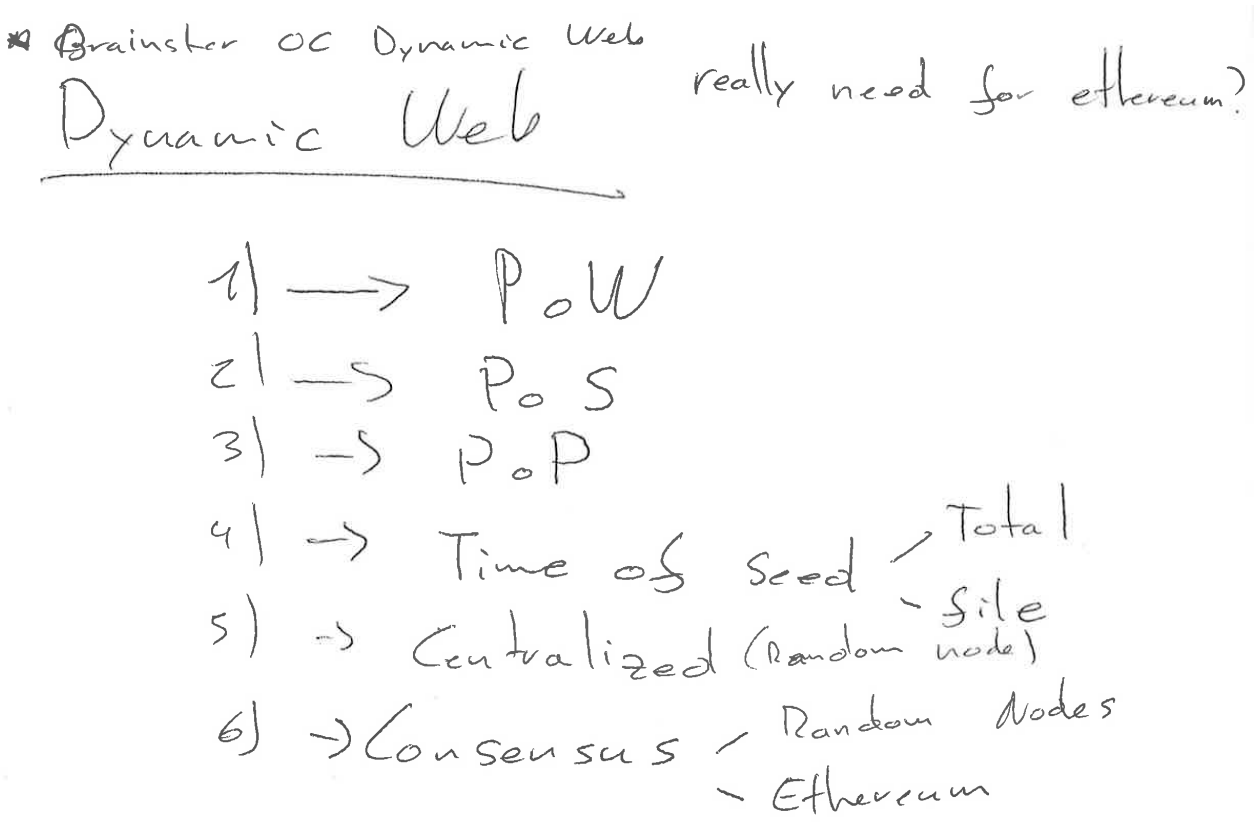


Figure 12: Brainstorming Dynamic Web

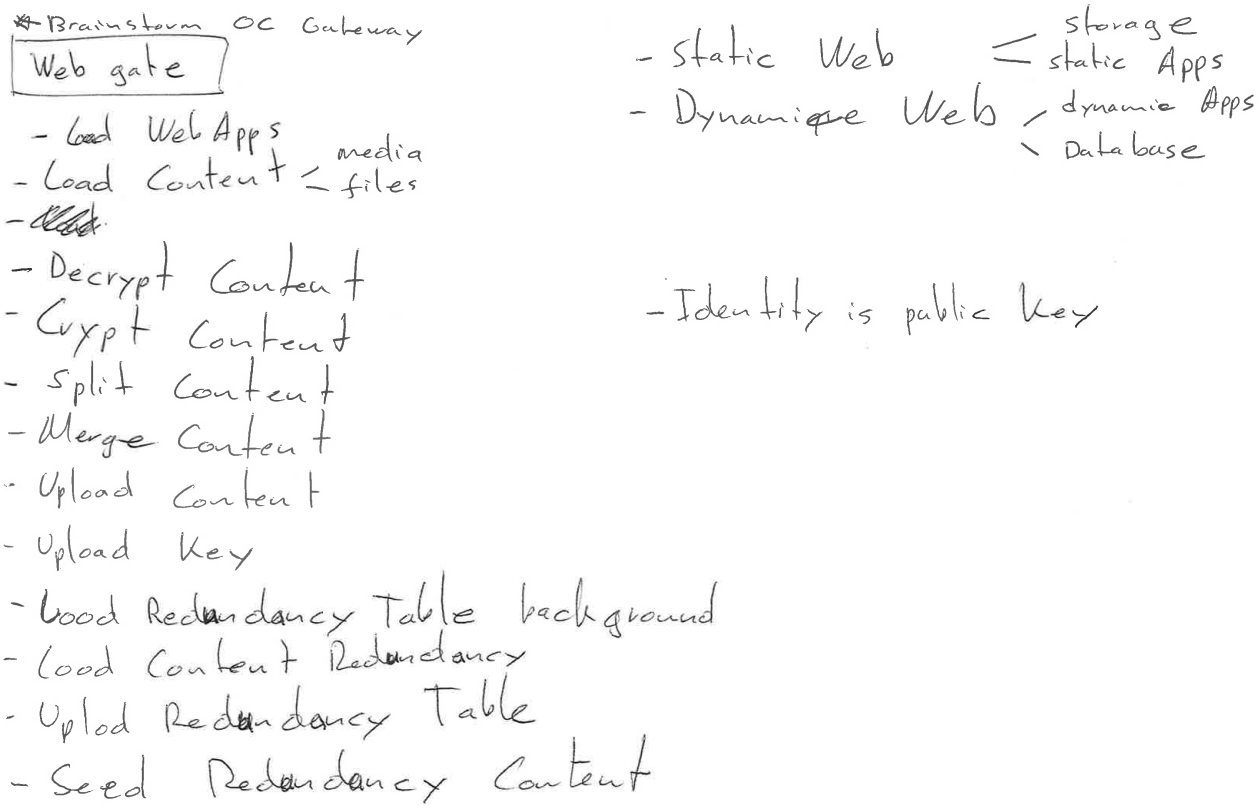


Figure 13: Brainstorming Webgate

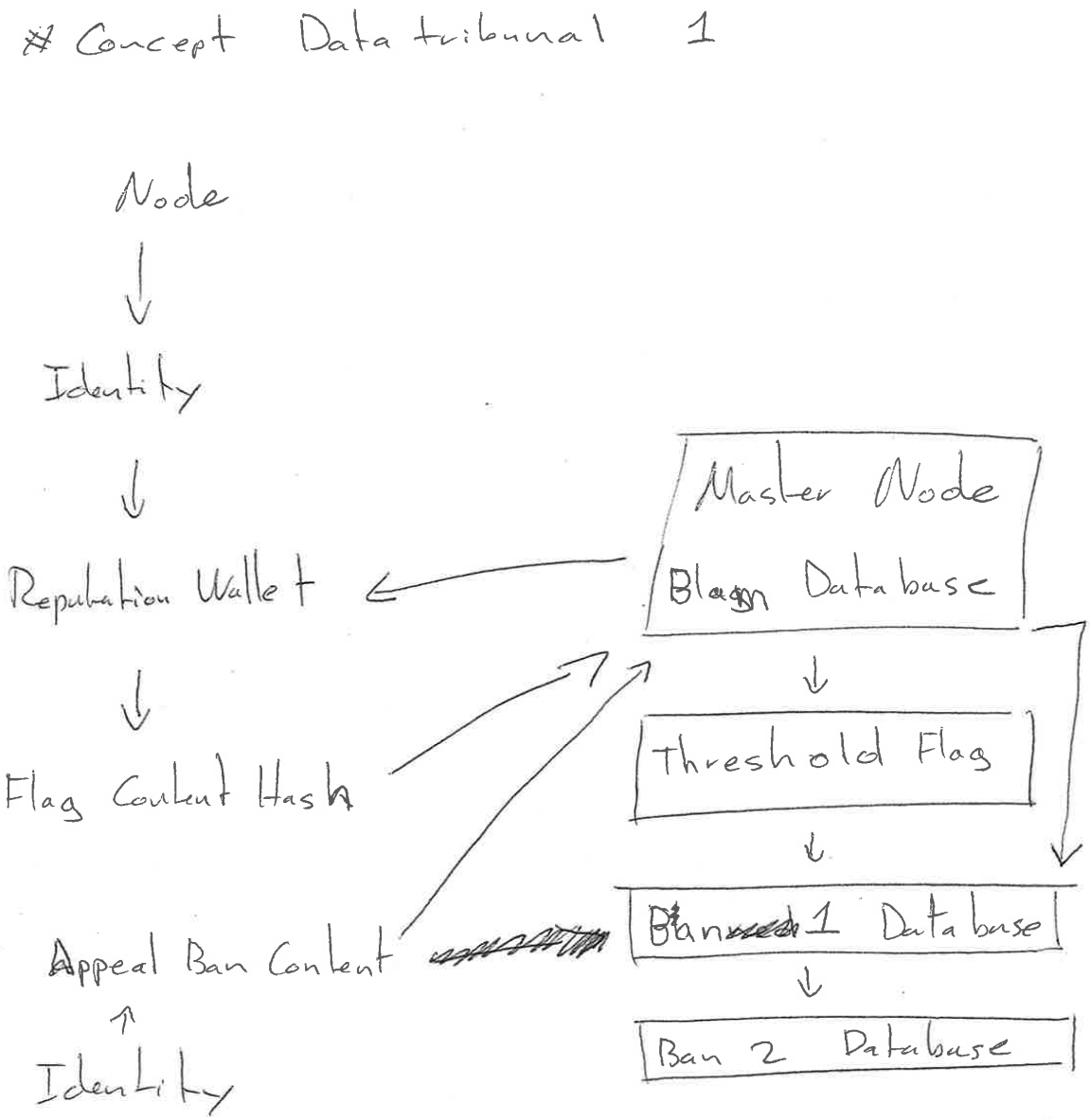


Figure 14: Concept data tribunal

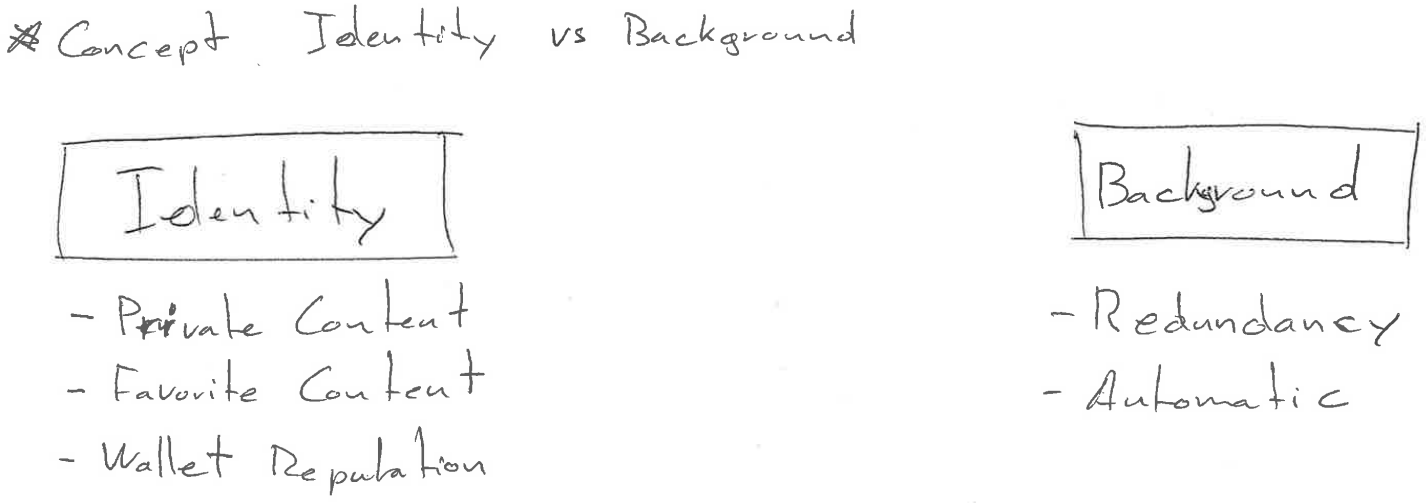


Figure 15: Concept identity vs background

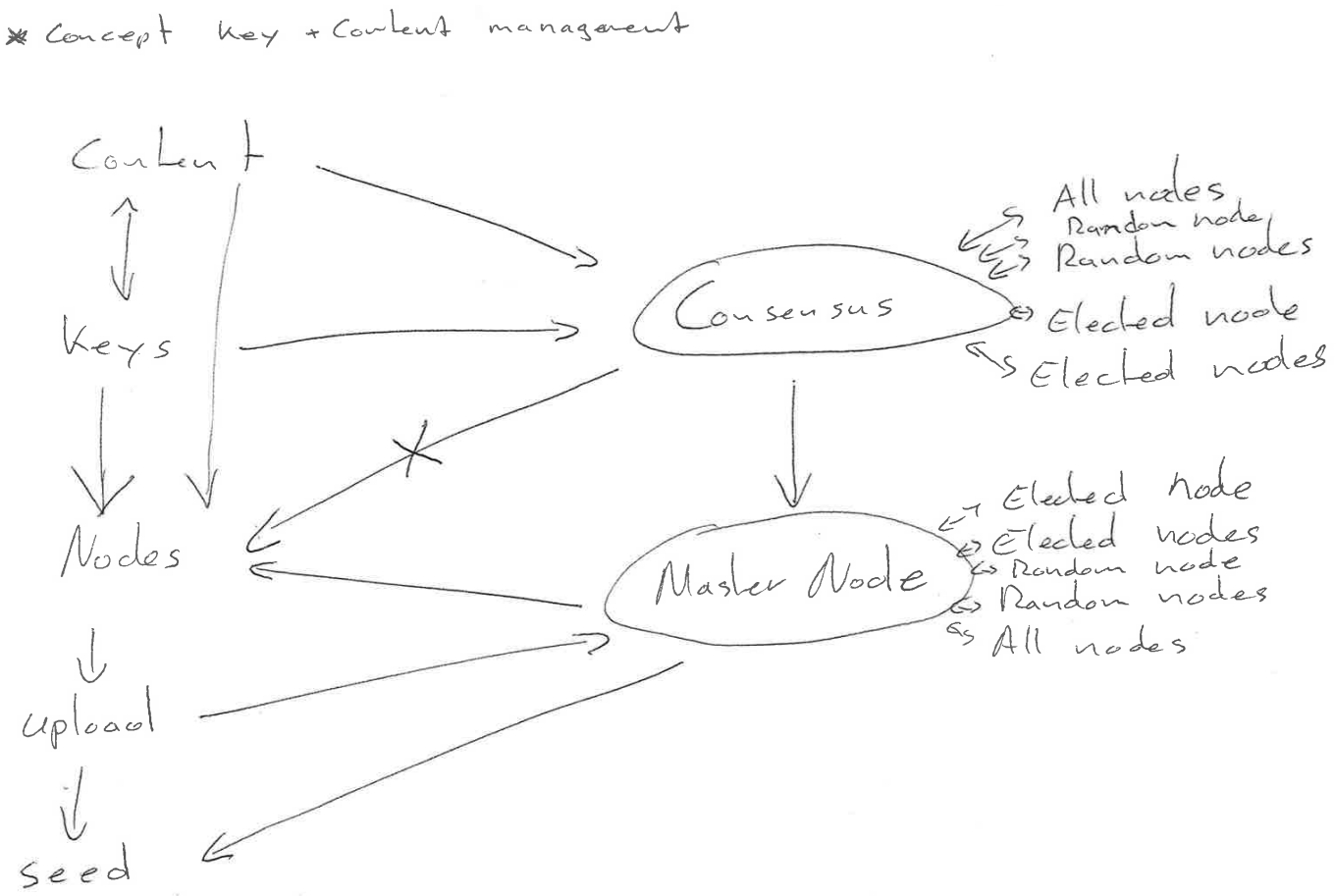


Figure 16: Concept content management

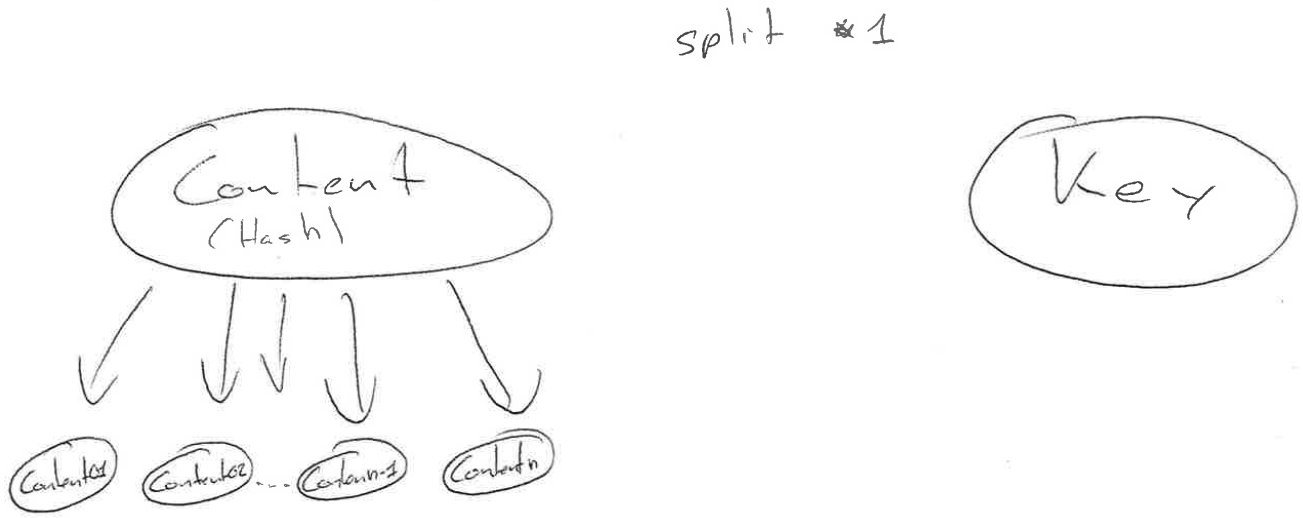


Figure 17: Concept data split 1

Figure 18: Concept data split 2

7.4 Schemes

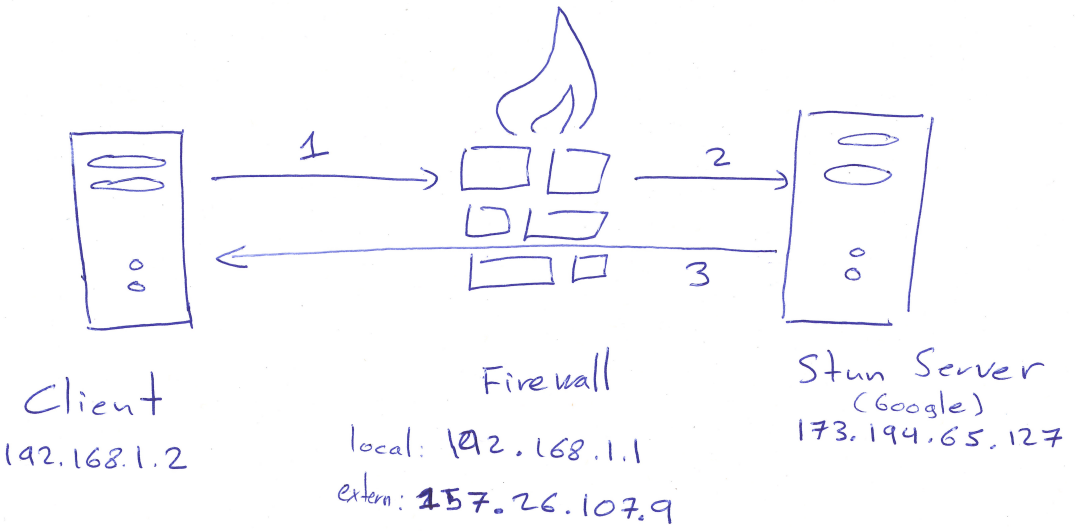


Figure 19: Serverless STUN Scheme

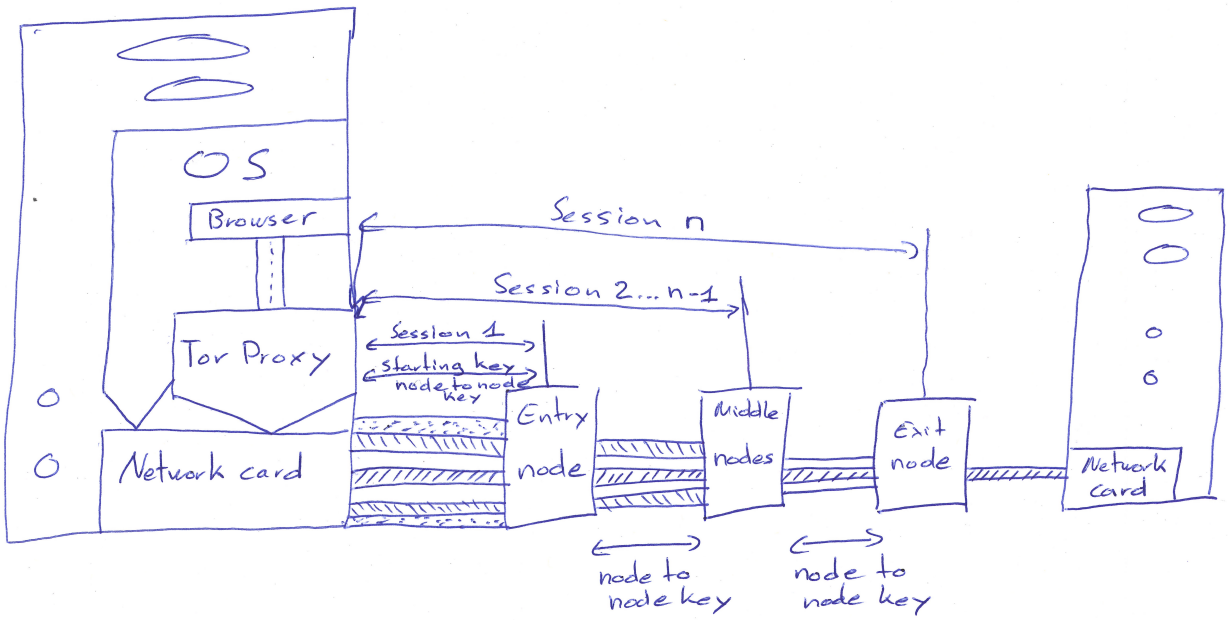


Figure 20: TOR Routage Schematic

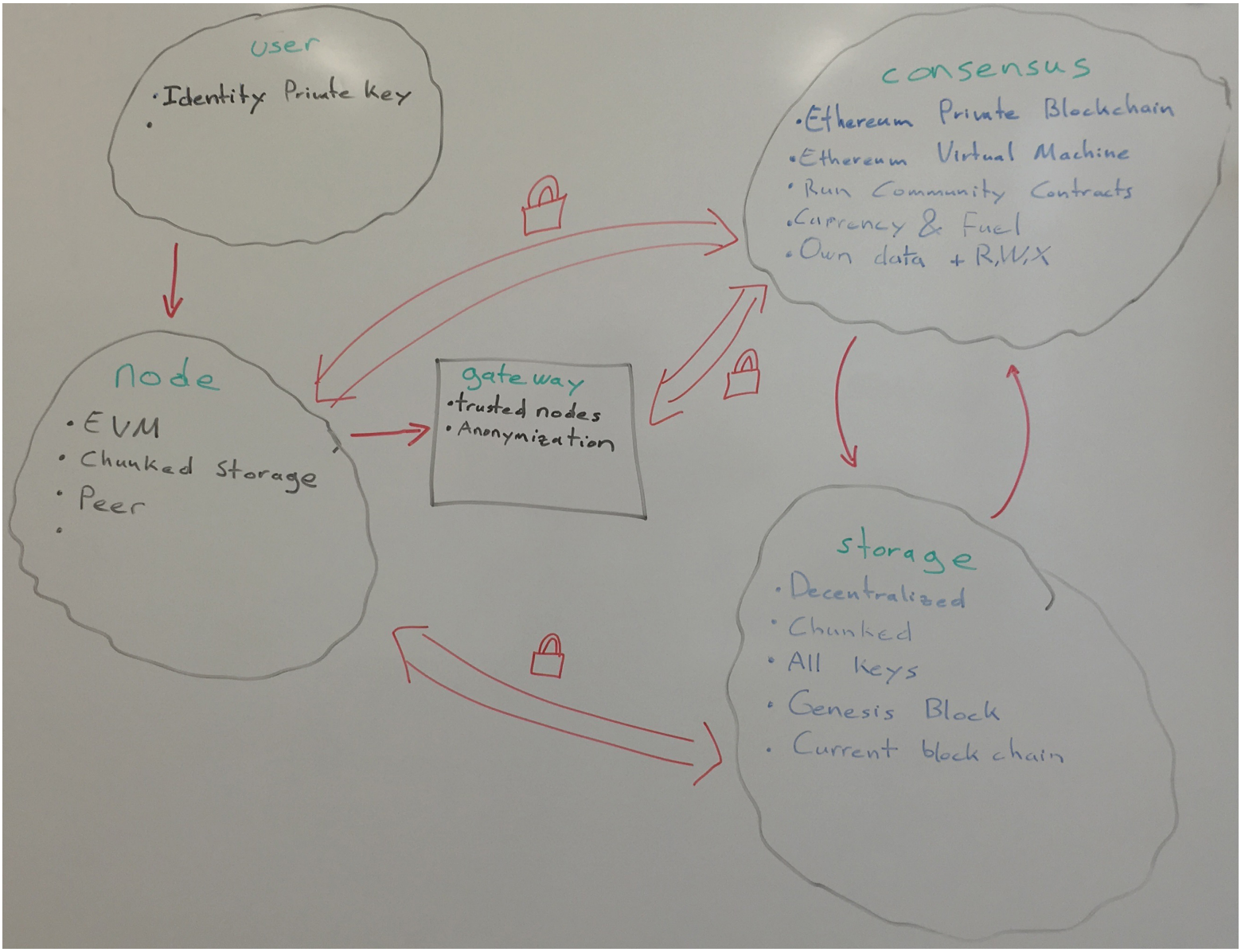


Figure 21: Schematic architecture version

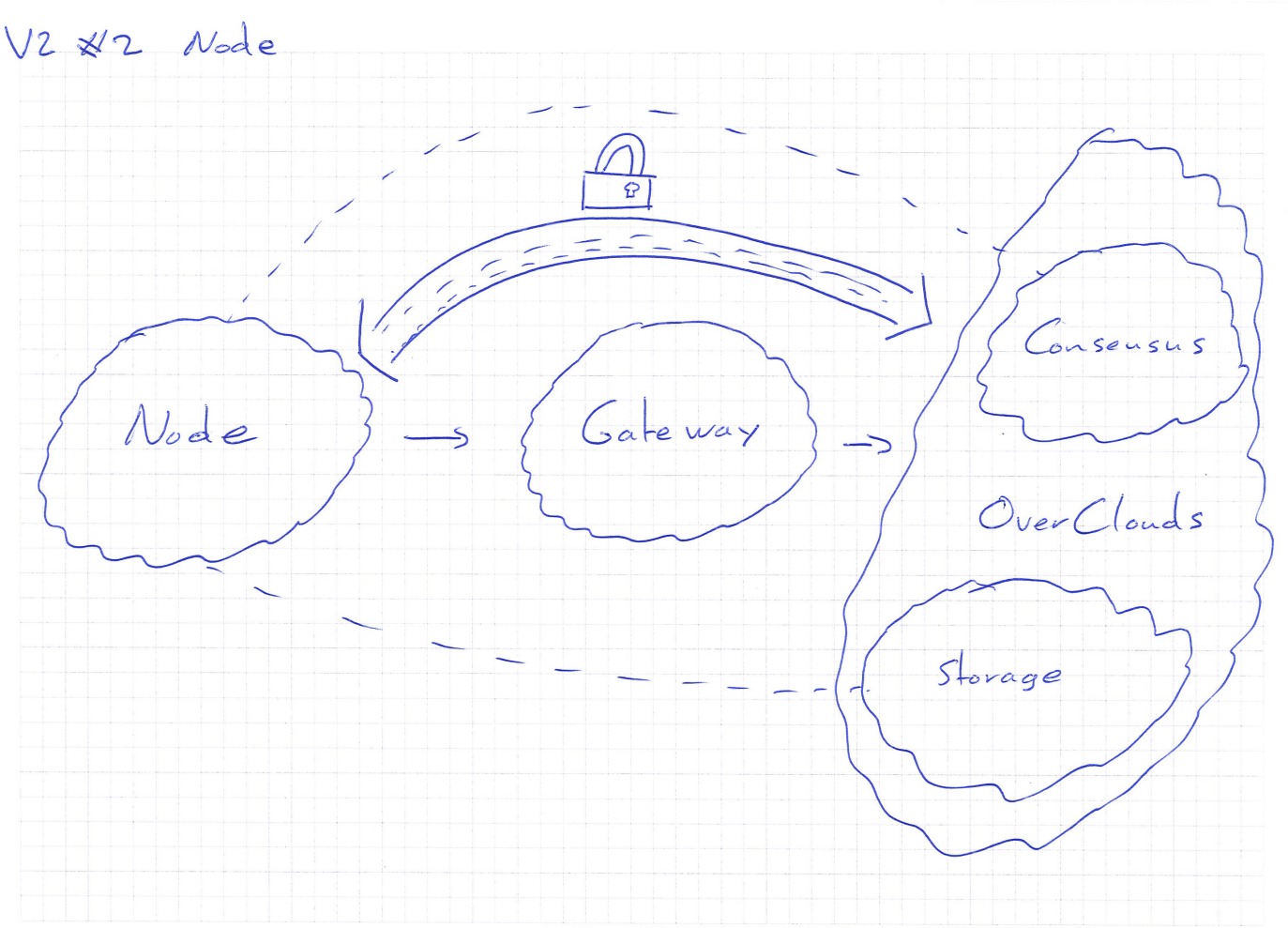


Figure 22: Latest schematic architecture version seen from a user (spring session)

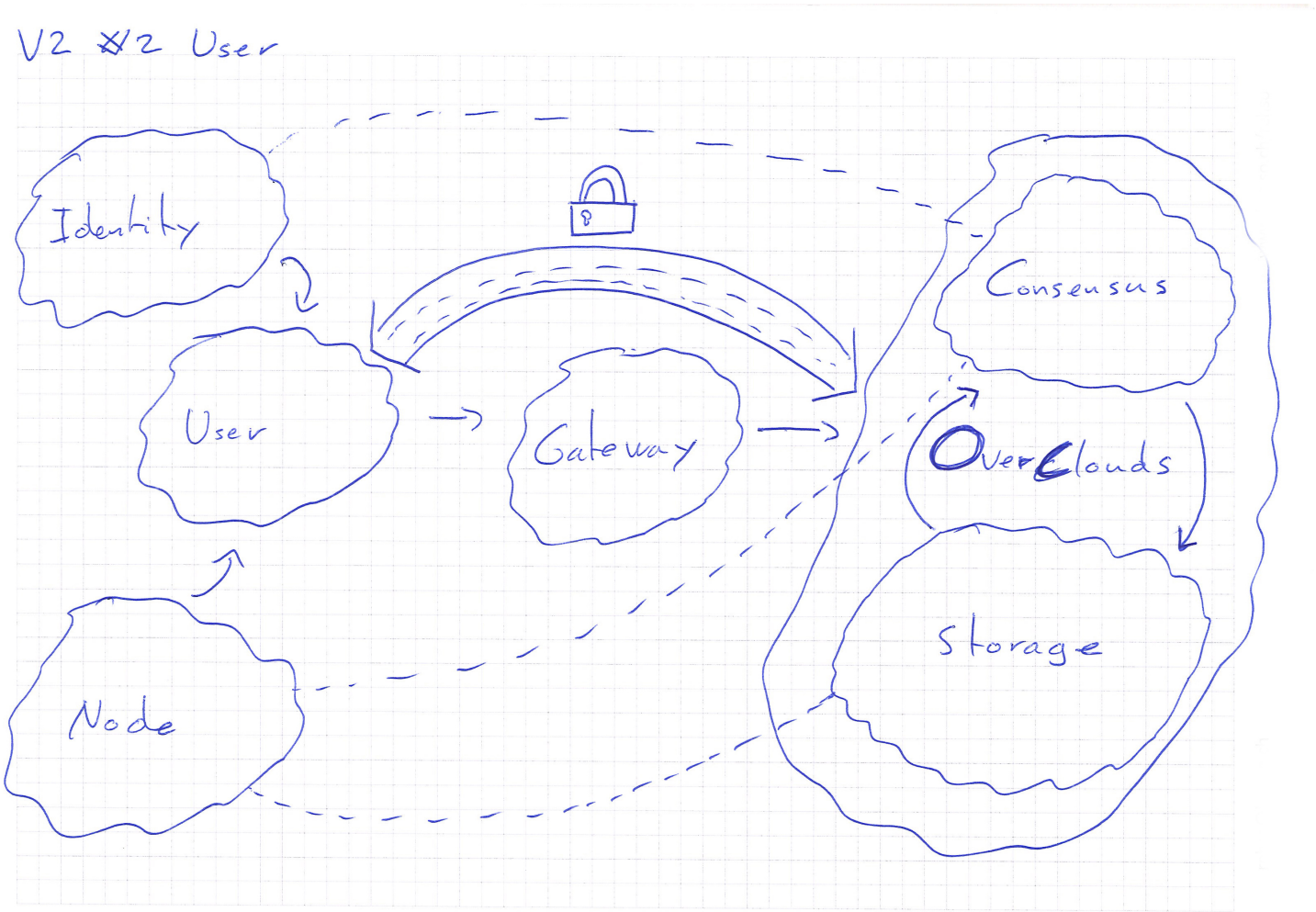


Figure 23: Latest schematic architecture version seen from a node (spring session)

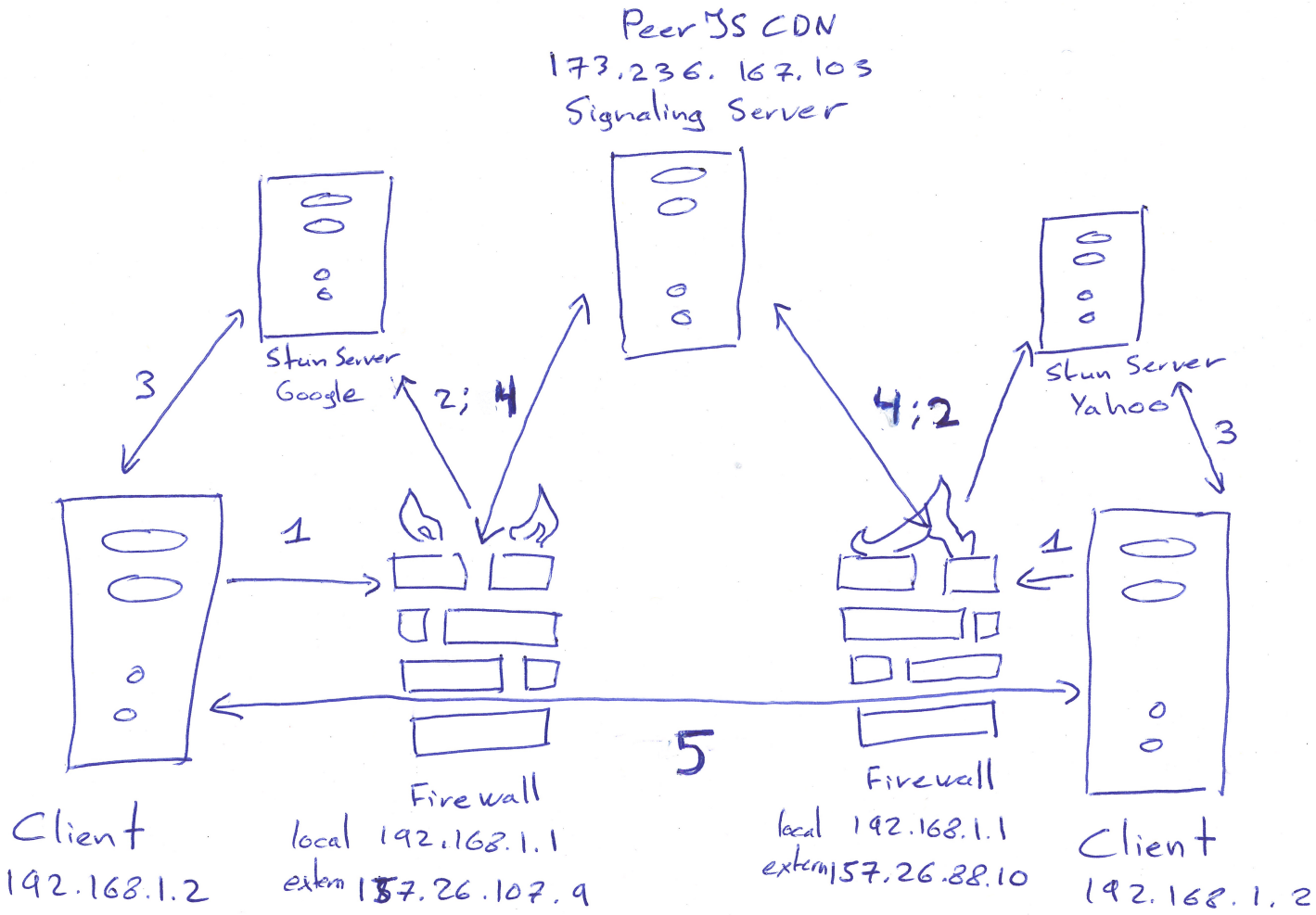


Figure 24: *Advanced Data Channel*

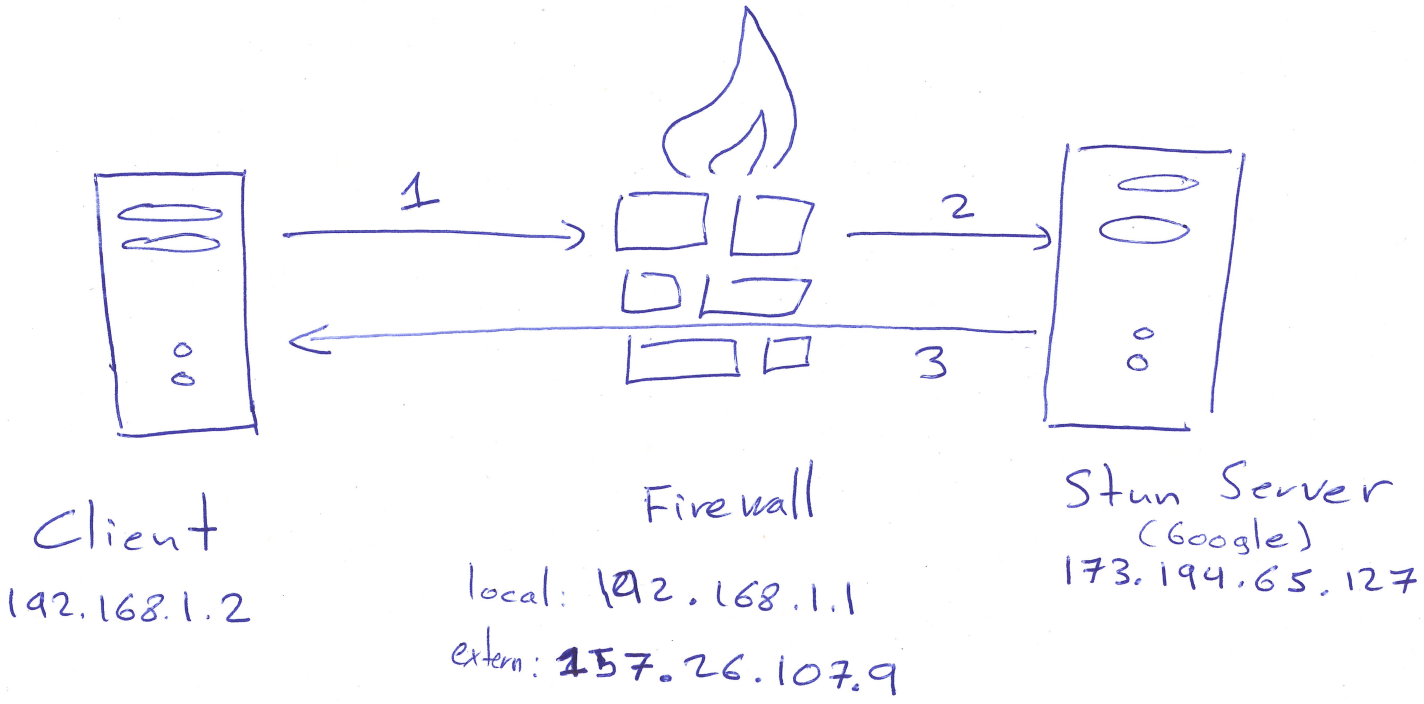


Figure 25: *Serverless STUN Scheme*

7.5 Screenshots

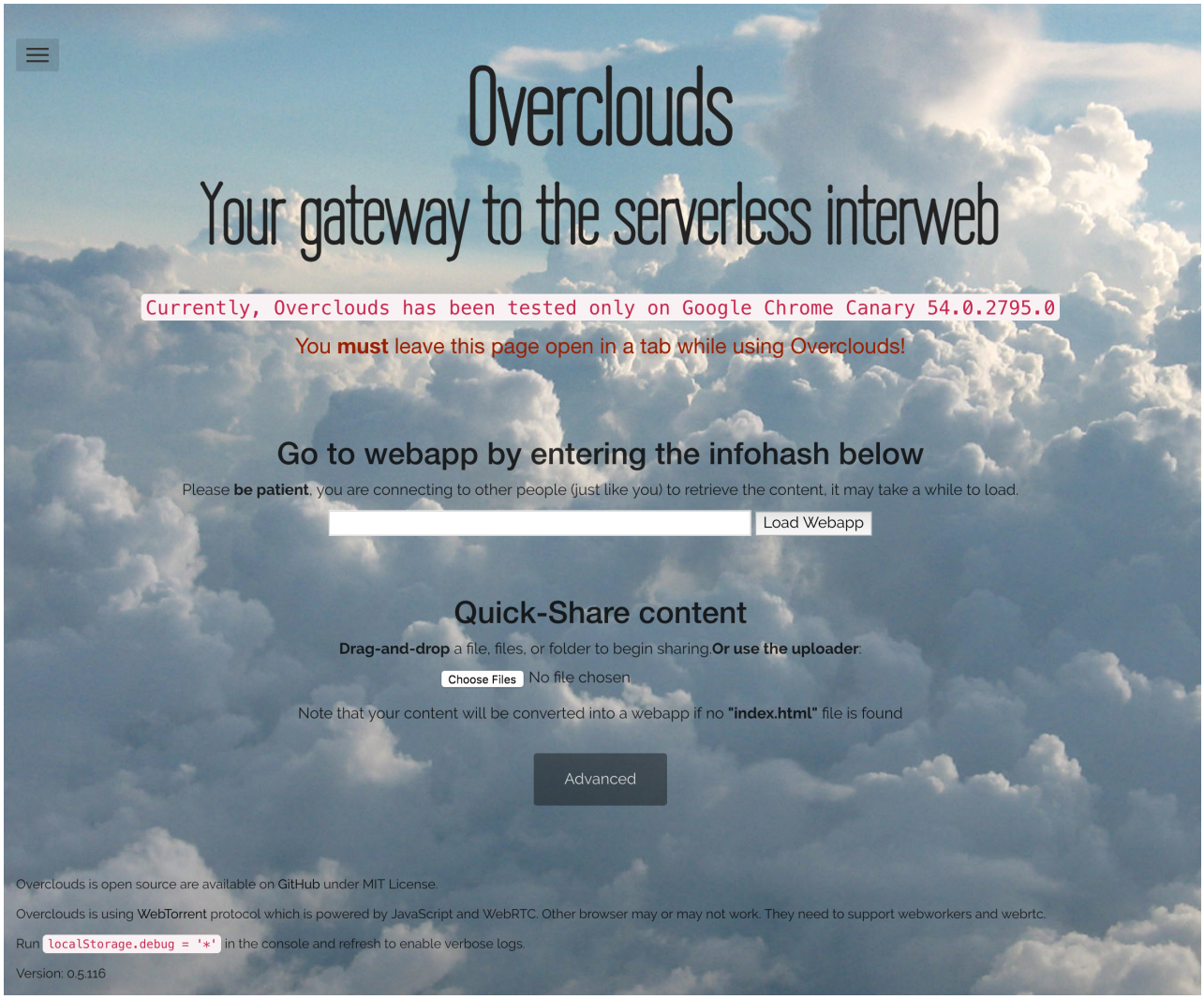


Figure 26: Screenshot from the main page of Webgate

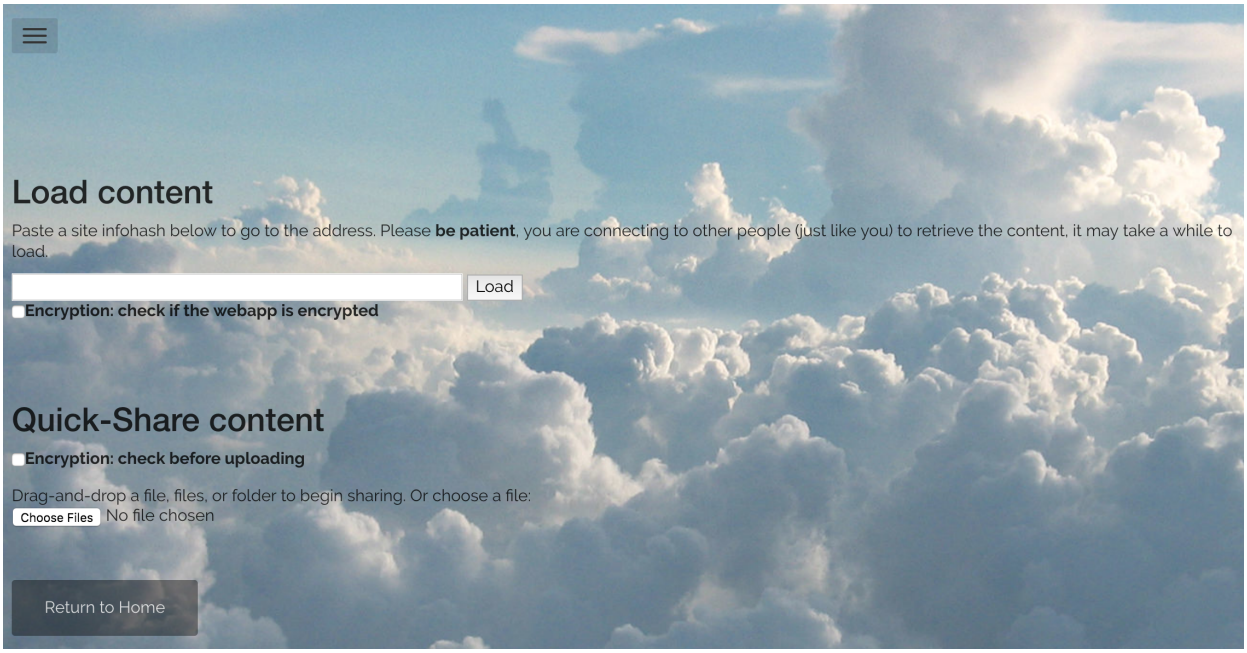


Figure 27: Screenshot from the advanced page of Webgate

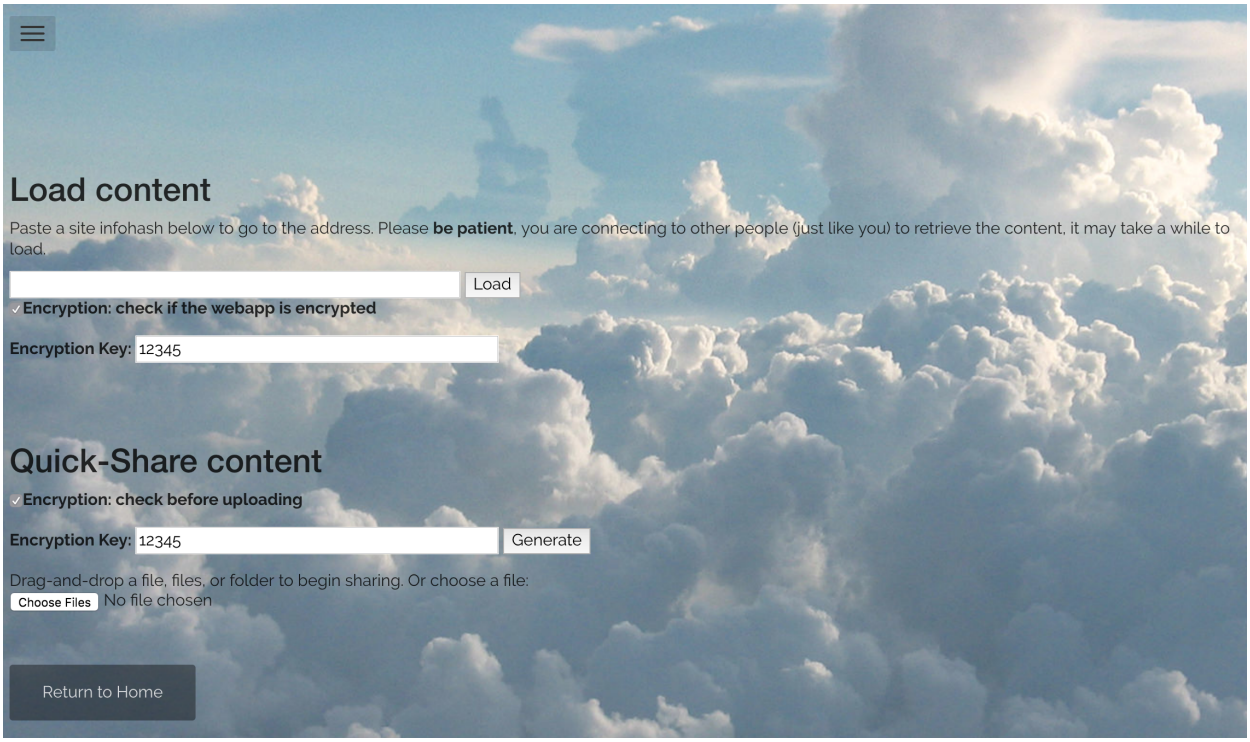


Figure 28: Screenshot from the advanced page with details of Webgate

7.6 Graphs

The graphs from the following figures have been made by **Dominic Tarr** [99]

Figure 29: *y-axis shows total time taken, lower is better*

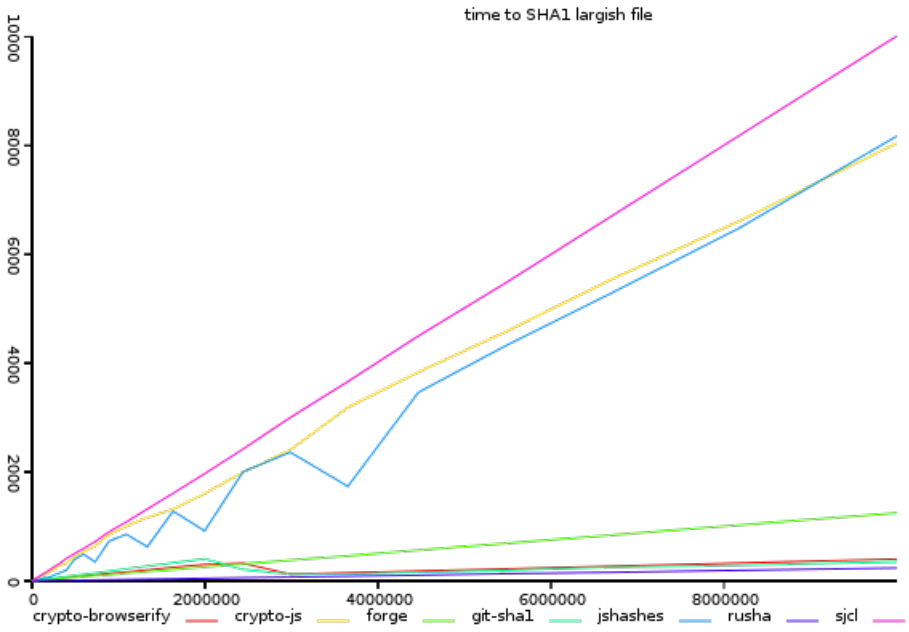


Figure 30: *y-axis shows size/time, higher is better*

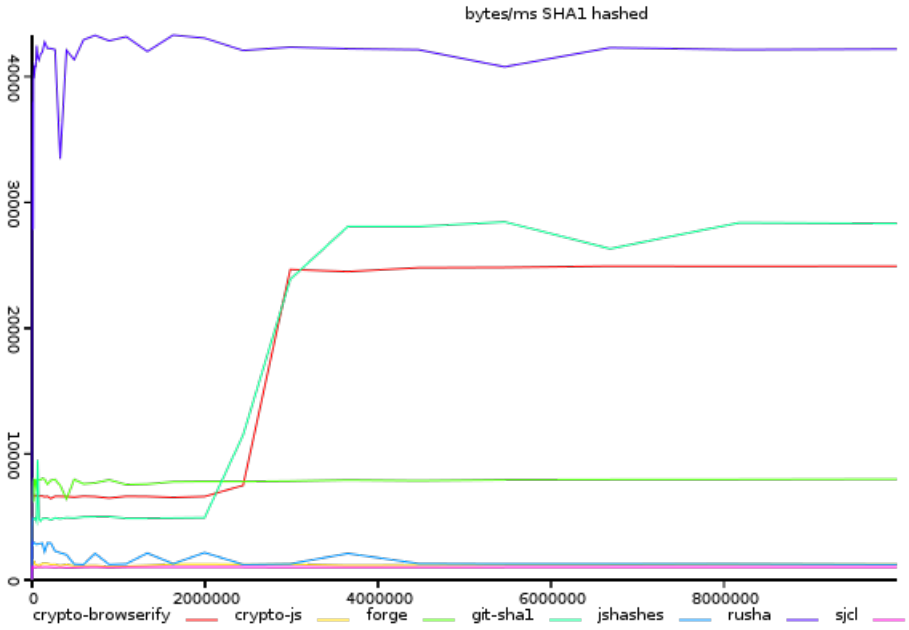


Figure 31: *y-axis shows total time taken, lower is better*

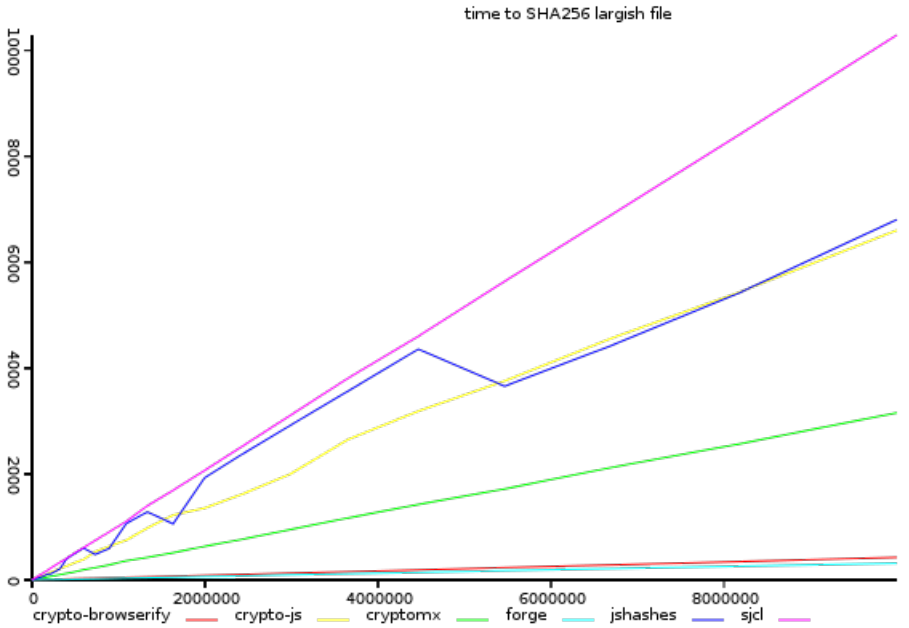


Figure 32: *y-axis shows size/time, higher is better*

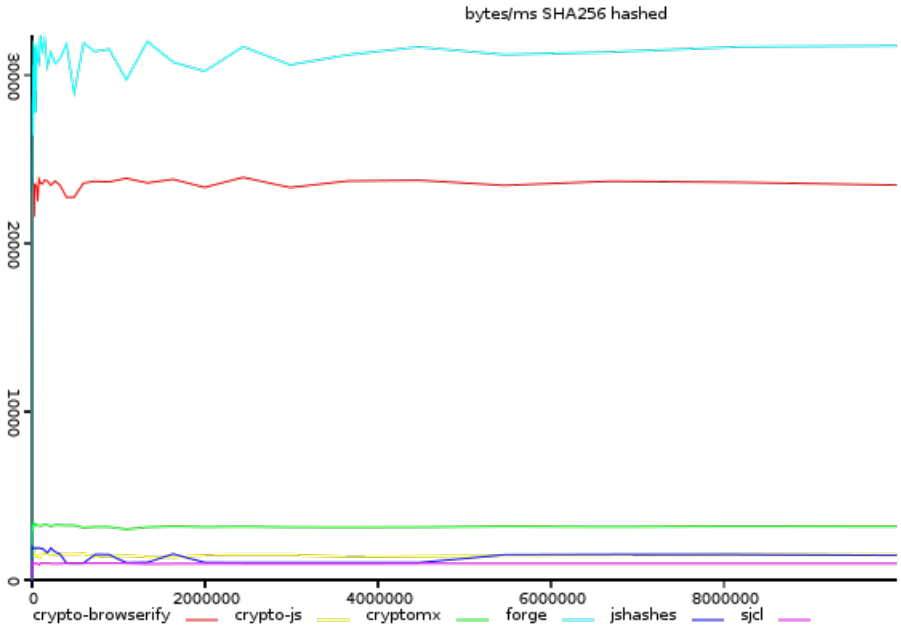


Figure 33: *y-axis shows total time taken, lower is better*

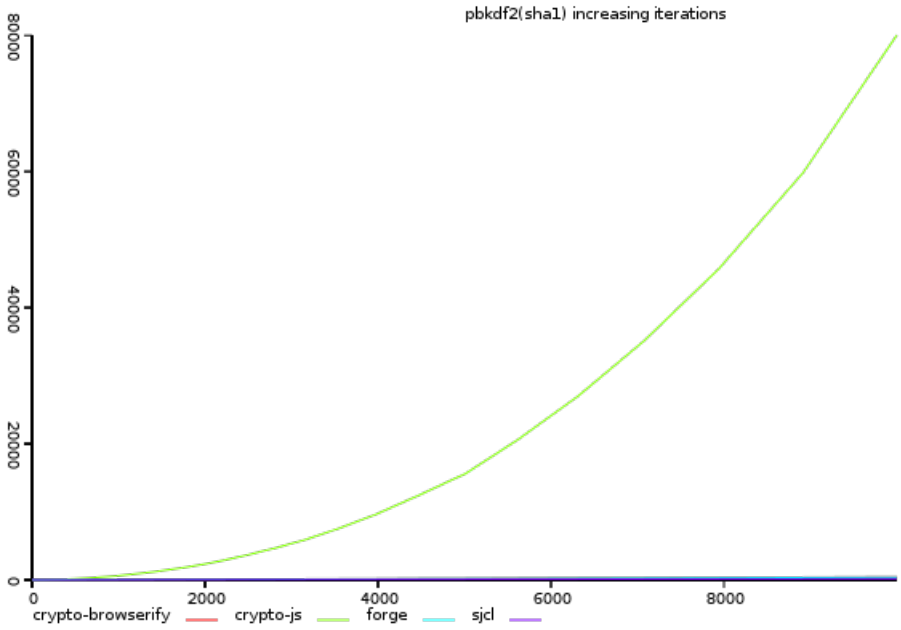


Figure 34: *y-axis shows size/time, higher is better*

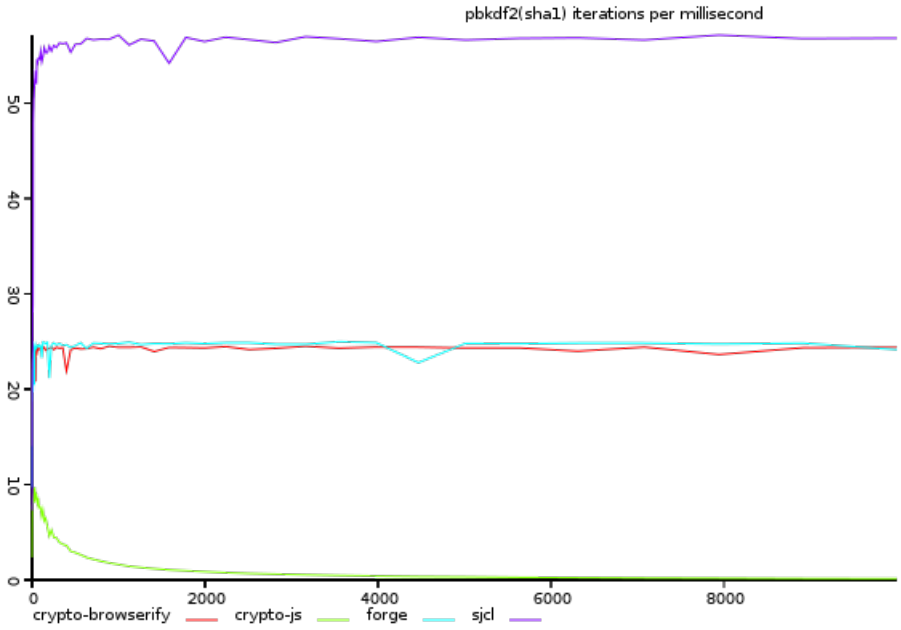


Figure 35: *y-axis shows total time taken, lower is better*

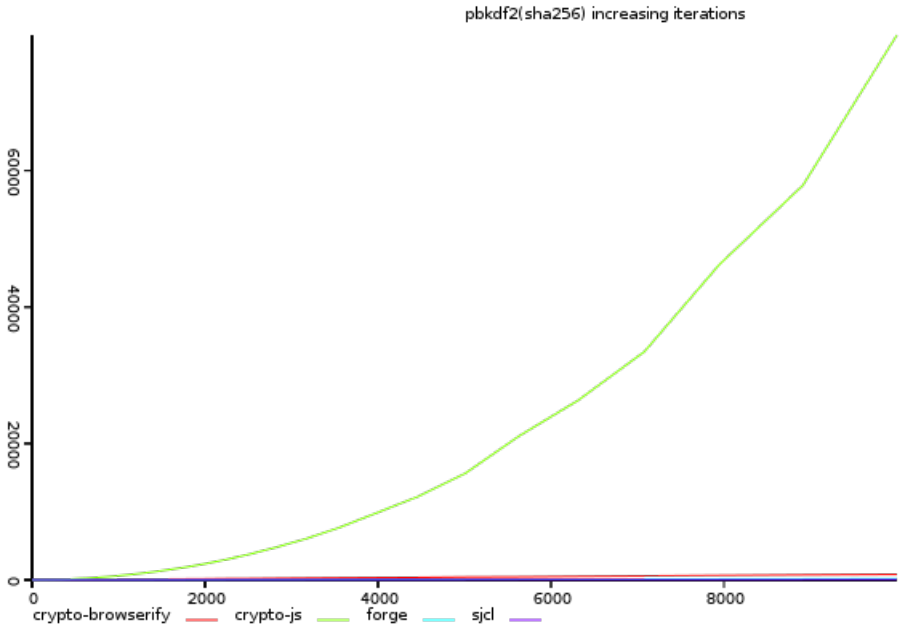


Figure 36: *y-axis shows size/time, higher is better*

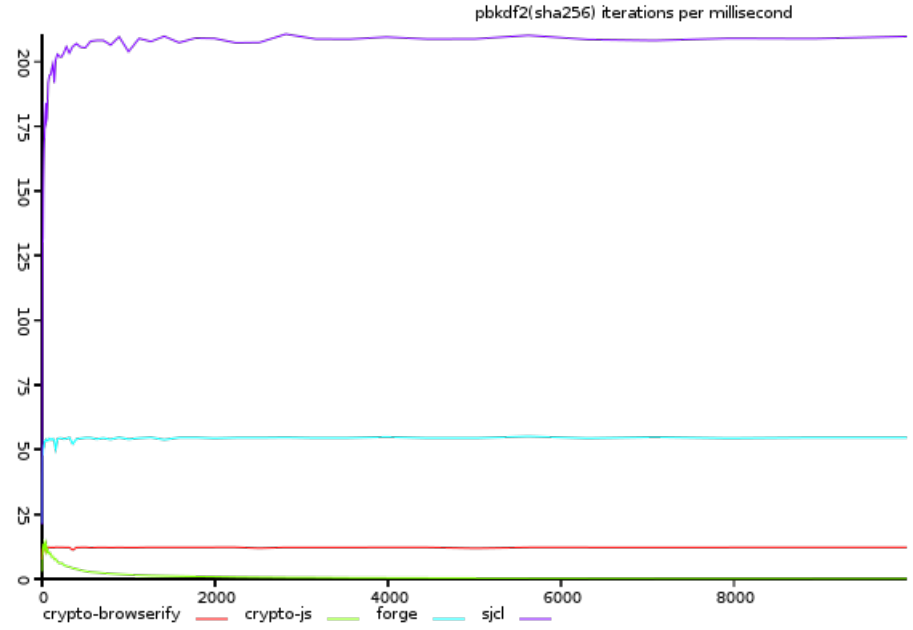


Figure 37: *y-axis shows size/time, higher is better*

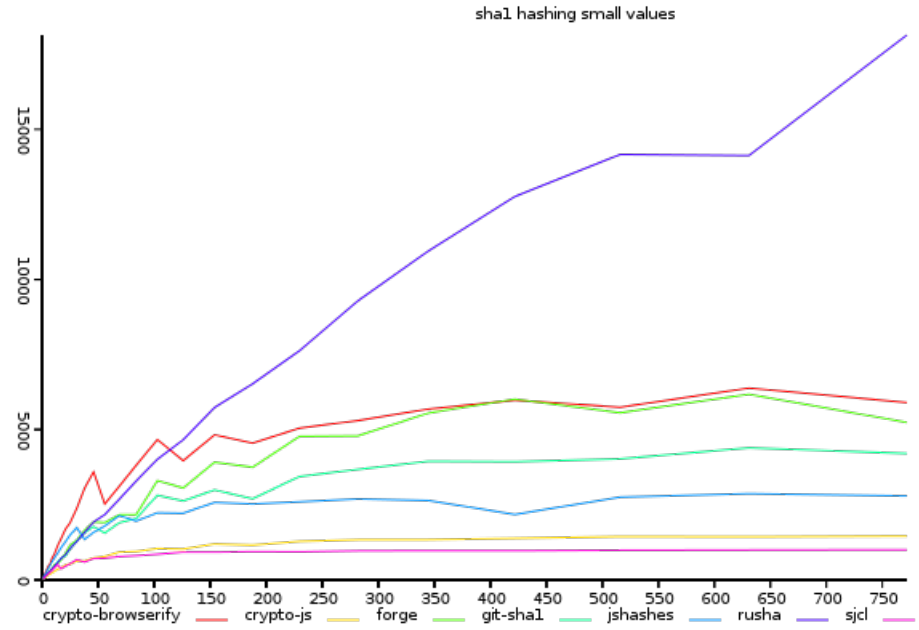


Figure 38: *y-axis shows size/time, higher is better*

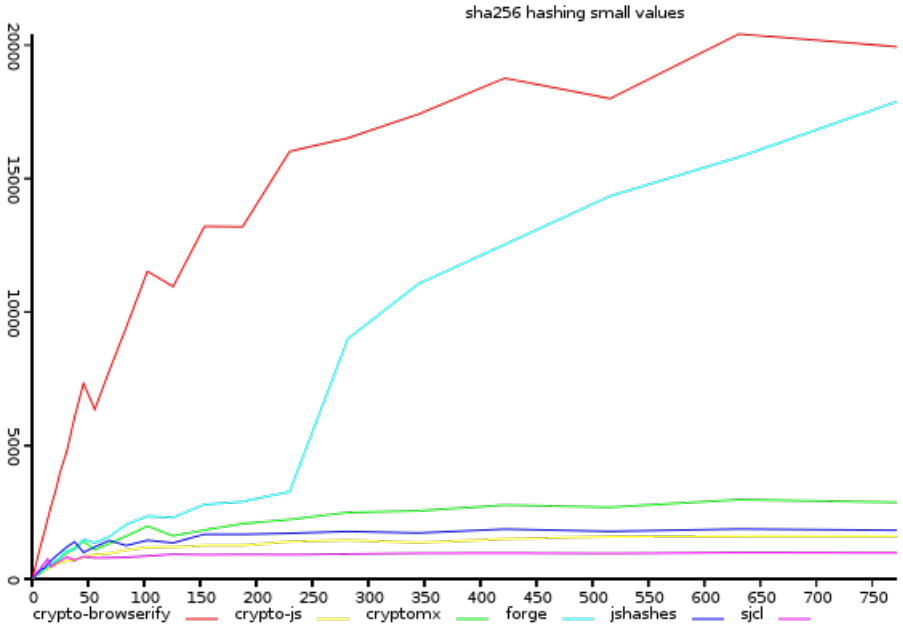


Figure 39: *y-axis size/time, higher is better*

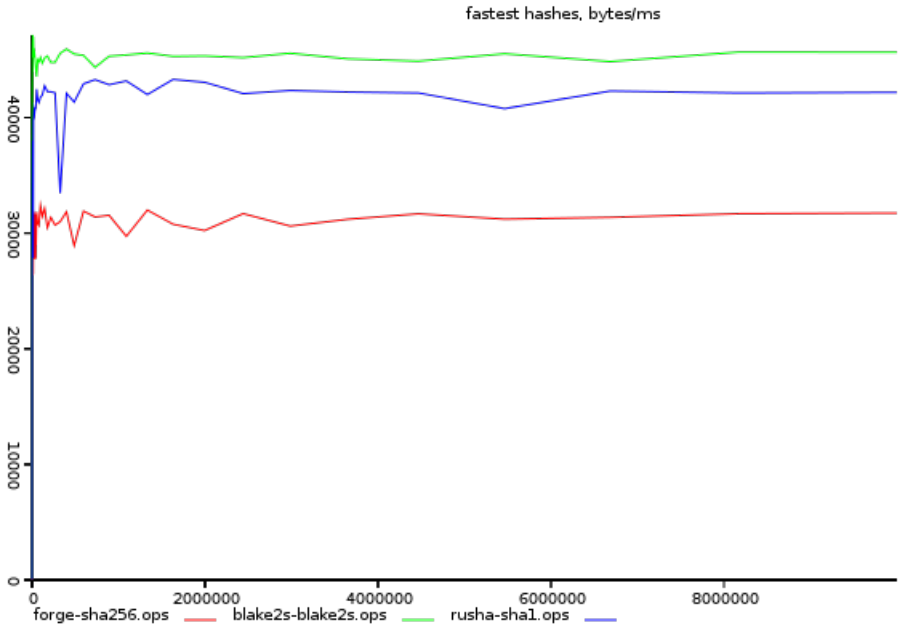
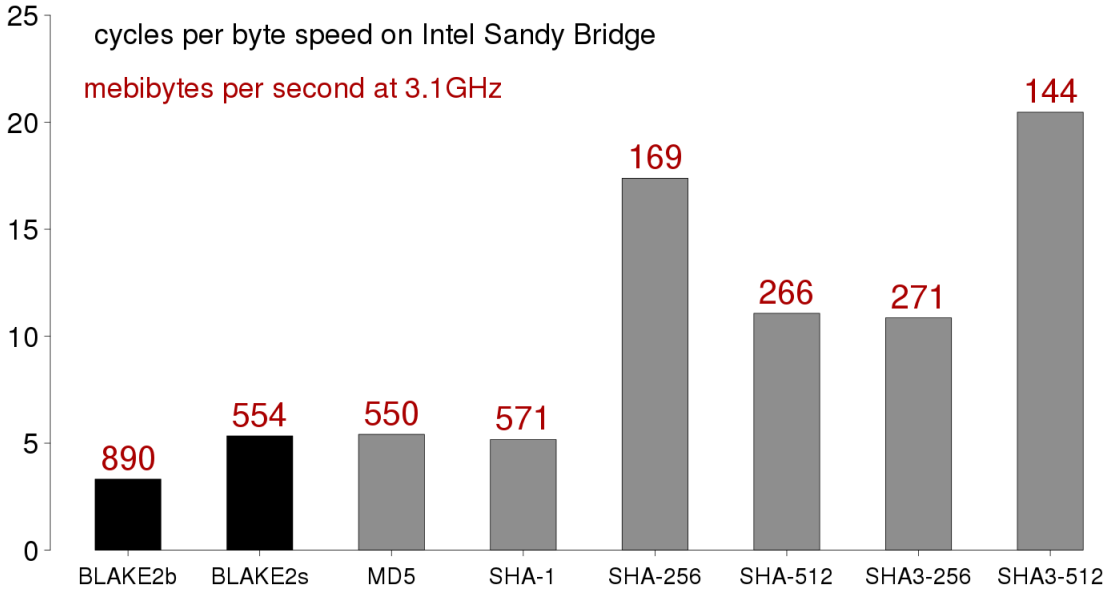


Figure 40: *lower is better*



7.7 Tables

The following tables have been made based on **Dominic Tarr**'s[99] benchmarks.
Table outcome must be interpreted in the following order.

- Awful
- Bad
- Ugly
- Meh
- Neutral
- Okay
- Nice
- Good
- Amazing

Table 1: Hashing 0-10MB Files /milliseconds based on figures 29, 30, 31, 32

Libraries	Sha1 (size)	Sha1 (hash)	Sha256 (size)	Sha256 (hash)
sjcl	Awful	Awful	Awful	Awful
crypto-js	Bad	Ugly	Meh	Bad
forge	Okay	Okay	**Amazing**	**Amazing**
crypto-browserify	Nice	Nice	Good	Good
crypto-mx	Neutral	Neutral	Okay	Ugly
git-sha1	Good	Good	Neutral	Neutral
jshashes	Meh	Meh	Ugly	Bad
rusha	**Amazing**	**Amazing**	Neutral	Neutral

Table 2: Hashing Small Files /milliseconds based on figures 37, 38

Libraries	Sha1 (size)	Sha256 (size)
sjcl	Bad	Bad
crypto-js	Ugly	Ugly
forge	Nice	Good
crypto-browserify	Good	**Amazing**
crypto-mx	Neutral	Okay
git-sha1	Okay	Neutral
jshashes	Meh	Meh
rusha	**Amazing**	null

Table 3: Key Derivation (pbkdf2) based on figures 33, 34, 35, 36

Libraries	Sha1 (time)	Sha1 (size)	Sha256 (time)	Sha256 (size)
sjcl	**Amazing**	**Amazing**	**Amazing**	**Amazing**
crypto-js	Awful	Awful	Awful	Awful
forge	**Amazing**	Nice	**Amazing**	Okay
crypto-browserify	**Amazing**	Nice	Good	Ugly

Table 4: Fastest Hashes /milliseconds based on figures 39, 40

Libraries	Sha1 (size)	Sha256 (size)	Sha3 (size)
rusha	**Amazing**	Neutral	Neutral
forge	Neutral	**Amazing**	Neutral
blake2s	Neutral	Neutral	**Amazing**