maidsafe: A New Networking Paradigm

Introduction
The maidsafe network is a truly distributed network which is scalable and self-healing, providing a highly-secure, highly-reliable method of data storage, retrieval and sharing. Data is located and relocated in such a way as to reduce bandwidth usage, minimise load on individual nodes and identify defunct hardware on the network.

Perhaps one of the most important aspects of this distributed network is the ability for users to validate who they are and with whom they are communicating. Self-authentication allows users to be free of corporate controls or risks regarding their data protection. This is achieved by users storing random data only they know the key of and password protecting that data for later re-validation; users essentially log into their own data.

Methodology
To achieve true full-distribution of the network, users are asked to donate a portion of their own disk space for use by the network. This avoids the requirement for centralised servers. Interestingly in the maidsafe design, any disk space you provide to the network is unlikely to store any of your own data.

System Layers
The maidsafe network can be considered as a three layer stack design as follows: 1) The Network Layer handles routing, NAT traversal and churn; 2) The MAID Layer implements all algorithms relating to self-healing (additional churn control), ranking, perpetual data, self-encryption and most importantly self or anonymous authentication; 3) The FileSystem Layer is the area of the system responsible for presenting the user with a filesystem from which to work with data. Initially this will be a read only file system and then be added to with the likes of a VFS or possibly a FUSE. The FileSystem and (most of) the MAID Layers are linked via the GUI.

FileSystem	GUI
MAID	
Network	

System Building Blocks
maidsafe technology effectively splits the user's computer into a processing/access device and resource managing devices referred to as clients and vaults respectively. The priority is that clients are simply access devices which can save and retrieve resources including data, messages and certificates, etc.

An important issue with distributed systems is the ability of a management or policing layer to ensure fairness and stability across the network. The maidsafe network utilises a ranking mechanism whereby vaults are continually assessed on their reliability and availability. The "best" vaults are used to provide such a policing layer called the supernode network, and hence vault churn is somewhat determinable and hence manageable. Clients are assigned addresses which are 8 bits longer than vault addresses. These longer addresses are never stored in nodes' routing tables and cannot be considered as an option for storage of [key,value] pairs, so client churn has negligible effect. Sets of three supernodes share responsibility for zones or network segments broadly corresponding to geographical locations.

Network Processes
A vault joining the network bootstraps off a supernode by providing it with a suggested random Kademlia address. The supernode alters the suggested address so as to achieve a balanced address space (hence optimising lookup iterations and scalability), ensure address uniqueness, and to assign the node to the appropriate network segment as indicated by its IP address. (As Kademlia uses XOR to define closeness of addresses, addresses can be assigned so that a group of nodes which are mutually physically distant are close in terms of Kademlia addresses. This is important as [k,v] pairs are stored on groups of nodes which are "Kademlia close" to each other, hence if these same nodes are physically distant then a large-scale partial outage of the Internet will not cause all the duplicate [k,v] pairs to be unavailable. Henceforth, "close" means "Kademlia close".)

The bootstrapping node then populates its routing table by doing a self-lookup after which the standard Kademlia bootstrapping is complete. However, at this stage the node asks its closest neighbours for the three local supernodes' addresses and registers with these supernodes. The three supernodes query the supernode network to receive the node's current ranking data, then regularly check and log the node's availability/status (amending its rank as appropriate) until the node re-appears elsewhere on the network (i.e. having left the network and bootstrapped again). Once a zone accrues an upper limit of nodes, the supernodes will promote the "best" three nodes within the zone to supernode status and divide the zone in two. The address space of the busy zone is bisected, and as the bootstrapping process ensures a balanced address space, the two new zones will generally contain similar numbers of nodes.

A client or vault detecting an error on the system, its peers or itself generates an alert which is sent to its three supernodes. The supernodes then resolve the problem and/or distribute the alert throughout the entire supernode network, from where it is pushed out to all the clients on the network. Interestingly the supernode network now has an overlapping database which can be used to store vault-specific information (such as rank, etc.)

Self-Encryption
An issue with today's encryption techniques is that a user's key, biometric data or passphrase is used to encrypt every data element, thereby exposing the key on every data element encrypted. In maidsafe, all data essentially encrypts itself; no information about the owner is stored with the data.

When a file is stored, it is split into many chunks. Each chunk undergoes bit-swapping with the other chunks; and is then encrypted using the hash of a different chunk as the encryption key. Each chunk is then renamed with the hash of its encrypted content and compressed. The encrypted chunks can now be saved in various locations across the network. The locations of

the chunks are stored in a "data map" (pre-encryption hashes and post-encryption hashes) which are concatenated to create a "data atlas" which itself undergoes self-encryption to create another data map, which also creates encrypted chunks. This data map (of the data atlas) can only be read by the original user once they are authenticated.

Perpetual Data

When a chunk is stored, it is not simply one copy sent to the network, but four copies which are stored in discrete network zones. Each node hosting a particular chunk is given details of the three other "partner" nodes relating to this chunk, and periodically checks the validity of these duplicate chunks. Initially, nodes will perform relatively frequent validity checks. However, as "trust" is built up, the validity checks can significantly decrease in frequency.

If a node detects that a chunk copy has been deleted or (more importantly) corrupted, it will send an alert to the appropriate supernode and will create a replacement copy of the chunk to be stored elsewhere on the network. The supernode will assess the veracity of the alert and will act accordingly; i.e. if the alert is valid, it will "punish" (by demotion or quarantine) or warn the affected node; or if the alert is false, it will punish the node which generated the false alert. Furthermore, because chunks' names are the hash of their contents, nodes can easily check themselves for data corruption and send an alert to their supernode about themselves. In this way, the network self-heals and has significant resistance to hackers and viruses.

Public Key Infrastructure

This also positively affects the PKI, as public keys are stored as encrypted chunks. This creates a fully-distributed, validated and self-regulating PKI. Using standard PKI rules, privately-signed data is checked with the public key of the ID used. Hence it is critical that the public keys are protected from malicious or accidental alteration or such a system would fail, as any cracker could essentially replace the public key and pretend to be the ID.

Again, this is where the benefit of hashing is apparent. The hash of ID keys is the chunk name and also the ID associated with the chunk (and key in the DHT). Chunk validity checking as described above, ensures the public key is valid. These anonymous chunks can sign non-anonymous chunks happily and then only they can delete them. Signing is built into the chunk name and cannot be retrospective, i.e. you could not create a key pair that creates the same resulting hash name, so fraud would be computationally near impossible. As the chunks are well-formed and valid, users can be assured that the public key they get is the key that was saved in that chunk and that it has not been tampered with. The maidsafe network also provides anti-fraud mechanisms by having all users copy all public keys from vaults they communicate with. Thereby fraudsters would generate system alerts by trying to replace this key. This gives the owner an opportunity to re-save the key onto the maidsafe network (reducing the risk of Sybil attacks, and with signed ID boostrapping reducing the risk of Sparticus attacks).

Duplicate Prevention

While it may seem counter-intuitive, this method of storing data is likely to yield large savings of hard-drive space. When a file is stored on the maidsafe network, the encrypted chunks are only stored if they do not already exist on the network. Two identical files (irrespective of filenames) stored by different users will yield the same encrypted chunks, and so duplicates will not be generated. Instead, the original four copies of each relevant chunk will now be listed on two users' data maps. Since only a small percentage of the data stored on today's hard-drives is totally unique, the potential for reclaiming massive amounts of storage space is high. Tests done at a local level back up this theory, as do tests done by EMC on a local duplicate removal system (in a single LAN) which showed disk space savings of 10:1 to 25:1. A more recent study by datamonitor puts savings from 4:1 to 89:1 with an average at enterprise level of 20:1 (i.e. a 95% saving), although only real world testing will yield a definitive answer.

Anonymous Authentication

Another vital aspect of the maidsafe system is anonymous authentication. A user authenticates himself/herself via three unique pieces of information; a username (U), PIN (P), and password (W). However, these three items are never transmitted or stored on the network. To login, the client initially retrieves from the network the data which is named as the hash(hash(U) + hash(P)). This data is then decrypted using U & P yielding a number (d) which was generated randomly when the user first created their identity. Next, the client retrieves the data named as hash(hash(U) + hash(P) + hash(d)) which is the encrypted data atlas. It is decrypted using U & W and yields the locations of all of the data maps associated with the user. (pbkdf2 is used to further encrypt the random number (d) in the ID packet. This is extended by a known calculable value (time since epoch in our case) to ensure even the ID hash is always different.) Thus the network has no need of servers maintaining lists of sensitive security details, and the user's anonymity is assured.

Future Trends

The future of this type of technology is a bright one. A self-managed digital ID which can spawn as many IDs as the system or the user decides it should (SAH512 hashing allows for 1.34×10^{154} results which equates to more available IDs than atoms in the universe) will create the potential for many interesting and innovative things such as: **Digital Coin** (a digital replacement for currency), **Digitally-Validated Voting**, **Solid State Stand Alone vaults** (low power, 'plug and play' devices with low processing ability attached to the network could expand its data-storage capacity automatically and easily), and **Read Only Operating Systems** (which would be immune to viral attacks).

Lessons Learned

Over the last few years, the maidsafe team have learned several lessons including: there is no "one size fits all" in software development choices of language, code review works very well, there's no room for ego in an innovation based organisation, if it's logical it will work, addons and additions not inherent in the design don't scale well (i.e. maidsafe technology inherently backs up data and allows sharing, not as an addon but as a logical design step). Above all, belief and determination with the right team can achieve what would appear to many to be miracles.