Ambisafe Server: Design Overview

revision 1.2

Andrey Zamovskiy
andrey@ambisafe.co

Abstract

Over the last 20 years there were a lot of attempts to build a decentralized internet-wide database. Bitcoin[1] has kicked off a new wave of disruption by becoming the first implementation that actually works. We believe that the key invention of Bitcoin is a consensus algorithm that enables a group of independent systems to reach an agreement on the state of account at any specific point in time.

Since the moment Bitcoin was launched we have seen hundreds of similar projects popping up. Distributed consensus technology has become mature enough to become a foundation layer for building a new type of transaction databases on top of it.

The skillset required in order to build a product on top of these systems is quite challenging. Typical cryptocurrency developer should have pretty good understanding of software architecture, distributed databases, cryptography, and economics. The majority of companies are failing to build reliable cryptocurrency products due to the lack of these skills among developers.

Design Goals

The goal of this project is to lower the minimal education level that is required for a company to build a cryptocurrency-based product. We do this by encapsulating cryptography and database reliability algorithms into a server software product.

In order to be useful for a wide set of applications the product should meet quite a challenging set of requirements:

- be asset agnostic
- support multiple security models
- it can not be run by a single company

Solutions

Here is how we address the problems mentioned above

[1] https://bitcoin.org/bitcoin.pdf

Decentralization

By providing the software that anyone can install in their datacenter as opposed to SaaS approach we address the following issues:

- service operators are not affected by the regulatory rules of software vendor jurisdiction
- decentralized nature of cryptocurrency network is preserved
- there is no central point of failure for independent markets
- service operators can easily add more components without participation or consent of SaaS provider

Supernodes

To avoid the necessity of implementation of currency-specific consensus algorithms for our software we rely on the so-called supernodes. Supernode in our terms is a SaaS service that provides an API endpoint for some specific cryptocurrency network. Examples are: blockchain.info, omnichest.info, and others.

We rely on them for the following:

- property state calculations, such as balances
- transaction broadcasting

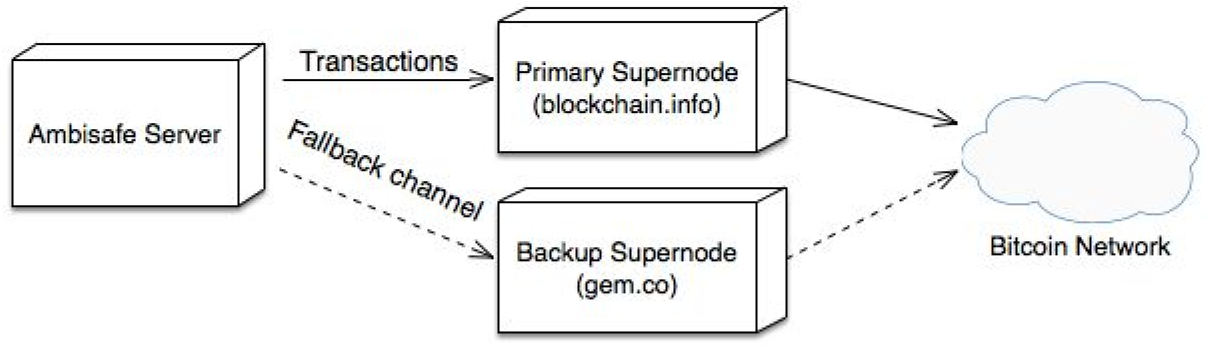
This approach should significantly increase the speed of adoption of the new currencies among ecosystem, because now it only takes to implement one interface[2] as opposed to integrating full network node into each platform.

Concerns

Reliability

Time to time websites go down and API providers are not an exception. We have addressed this concern by implementing a fallback mechanism. We have a backup supernode for each currency, so if the primary one goes down, another one will pick up operations. Since service reliability is the primary value proposition of supernode operators, having 1 backup supernode per currency should be enough.

[2] https://ambisafe.atlassian.net/wiki/display/AS/Developer+Guide


Transactions
Fallback channel
Ambisafe Server
Primary Supernode (blockchain.info)
Backup Supernode (gem.co)
Bitcoin Network

Security

Disaster scenario	**Recovery scenario**
Supernode fakes account balance	Service operator receives complaints from users about fake balance or inability to send transactions. After investigation service is switched to the backup supernode. **Money lost: 0**
Supernode refuses to broadcast transactions	Service operator receives complaints from users about inability to send transactions. After investigation service is switched to the backup supernode. **Money lost: 0**
Faking broadcast of transactions	Service operator receives complaints from users about transactions not arriving to destination. After investigation service is switched to the backup supernode. **Money lost: 0**

Containers

In multi-sig[3] security schemas account security is achieved by having several required co-signers for the account. In a world that is perfect from a cryptographer's perspective those co-signers should store their private keys on some device that they own and never transmit them over the network.

In a real world, however, majority of users are not capable of securing their keys properly and tend to loose them quite often. Therefore, it became a normal practice for cryptocurrency wallet service providers to host user-owned keys encrypted with password that is only known to user. Every time the user signs into the system, their container is downloaded from the service provider database and decrypted at user's device. Assuming that user password is never known

[3] http://bitcoin.stackexchange.com/questions/3718/what-are-multi-signature-transactions

to a service provider, the level of security is equal to the perfect world case, explained in previous paragraph.

Ambisafe server contains API and SDK that allows to create, store, and decrypt containers described above.

Concerns

Safety of Cryptographic Algorithms

We use AES-256 block cipher in CBC mode with padding for symmetric encryption and 1000 iterations of PBKDF2 of SHA512 of password for key derivation. Random number generator that is used to produce salt and initial vector is use-case specific.

This combination is the one that is recommended[4] and considered secure by cryptography professionals.

Security Schemas

Each account has one security schema associated with it. Security schema defines security model for that account. This includes multisig M and N, which keys are allowed to sign and other rules.

M - minimal number of required signatures.

N - total number of public keys that were used during address generation.

Examples of security schemas that are built into distribution:

Simple - classic 1 signature address

Wallet4 - 3-of-4 multisig account

Security schemas are loaded and configured in a modular manner. By making security schemas modular we are giving service operator flexibility to implement any security process, specific for their business. It can involve hardware security modules, oracle services and 2 factor authentication.

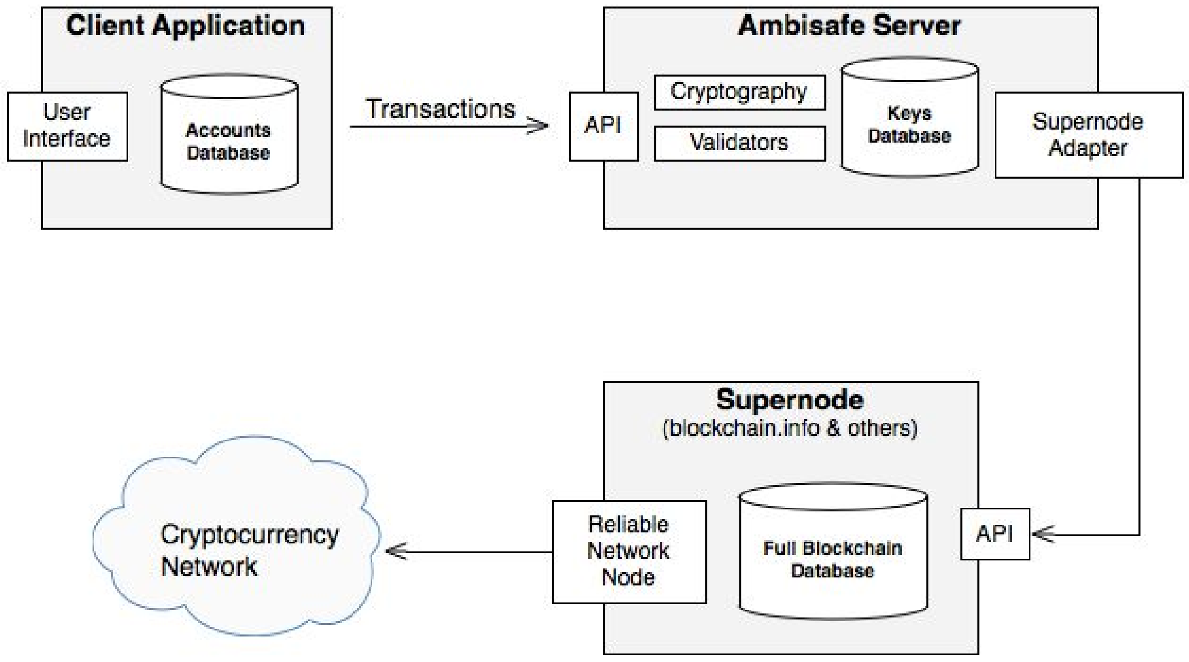
Asset Plugins

Asset(currency) plugin defines address and transaction formats, signing algorithms, and validation rules. Since address format is something network-specific, asset plugin will normally be set up together with supernode adapter for that asset(currency).

[4] http://crypto.stackexchange.com/questions/2251/how-secure-is-aes-256

Assets are implemented independently from supernode adapters because some cryptocurrency networks have more than one currency circulating on top of them. Examples are OmniLayer[5] and OpenAssets[6] for Bitcoin. As a consequence of this solution it is theoretically possible to replace underlying network for bitcoin 2.0 asset by making a couple of simple changes in configuration.

Typical Transaction Workflow



5 http://www.omnilayer.org/
6 http://coloredcoins.org/