THE UNBREAKABLE CODE

NAVTECH WHITEPAPER 2016

BETA RELEASE v0.9

INDEX

EXECUTIVE SUMMARY

Financial privacy is one of the cornerstones of any democracy. As valuable to a democracy's function as the freedom of speech. If fact, we argue that in a monetised society, without financial privacy, there can be no freedom of speech.

Nav Coin's Navtech system provides users with a simple way to protect their financial privacy. It is easy to use, and transactions arrive at their destination within minutes of being sent. Just tick the box and send the coins, Navtech handles the rest.

Behind the simplicity, Navtech is powered by cutting edge technology that processes and anonymises transactions using a secondary blockchain known as the Subchain.

Fast confirmation times and borderless transactions make Nav Coin the ideal vehicle for modern international private value transfers.

INTRODUCTION

Many entities around the world routinely discriminate, blackmail, imprison or even execute people for holding beliefs which go against their religion, policies or ideals.

An individual's finances are heavily coupled to how free they are to exercise their other rights.

In some societies a journalist can't work for a news outlet that writes content opposing the government; people of differing religious beliefs can't publish their arguments; an employee can't work for a group that promotes gay rights; a concerned citizen can't donate or subscribe to causes he supports without fear of punishment.

These voices are silenced because they can't operate with financial anonymity for fear of their affiliation provoking violence towards themselves or their loved ones.

These are dramatic examples, but they are just the tip of the iceberg. Regular people have their everyday choices influenced by the fact they don't have the guarantee of financial privacy. Whether it's who you work for, which products you buy, or which organisations you donate to, everyone's directly or indirectly affected.

Nav Coin aims to give people a democratic choice to make their financial transactions Private.

PROBLEM DEFINITION

Blockchains by their very nature are a public ledger. This is a vital part of what makes blockchain based software so powerful. All transactions on the Nav Coin network are recorded in the public blockchain so the network can validate them as being correct. The problem with this model concerning privacy is that it is possible to easily link the sending and receiving address with any of the publicly available block explorers.

Cryptocurrencies in general do offer some anonymity for their users because you don't have to register your name to your wallet address. You are also able to generate a new wallet address for each transaction and even connect to the blockchain via services which hide your IP address like Tor. However, the underlying problem remains, there is always a transactional link from sender to receiver. This exposes an inescapable vector where third parties can attempt to invade your financial privacy by making the connection.

There are existing services which attempt to break this transaction chain in various ways. Some offer coin mixing services, coin joining services and other more complicated methods of obfuscating the sender and receiver's details. The main issue with most of these methods is that they are either eventually traceable through the public blockchain (if you have enough computing power), or they rely on insecure and unreliable means (like a database) to keep track of who is supposed to end up with the coins, or both!

Using a database to keep track of the transactions being anonymized is in direct conflict with the decentralised and immutable nature of blockchain technology. When a database is introduced to the system, it means that someone can fake transactions and extract money by simply adding a row to the database table which keeps track of who to send money out to. There are many known attack vectors in databases and malicious actors routinely compromise them.

Secondly the system becomes vulnerable to data loss. If the database were somehow corrupted, destroyed or otherwise disabled, the whole system has to roll back to the last backup point which can mean thousands of dollars of missing transactions.

We have set out to create a transaction anonymizing system which truly breaks the link between sender and receiver while not relying on any type of database or centralised service.

HIGH LEVEL SOLUTION

The approach we have taken to disconnect the sender from the receiver uses multiple techniques which will be further outlined in the Solution Details section. The main technique that we have employed is to use a second blockchain we call the Subchain.

Instead of sending NAV directly to the receiver, the wallet encrypts the receiver's address and sends the transaction to one of the addresses provided by the randomly selected processing server. When this server receives this transaction, it creates a transaction of arbitrary size on the Subchain which it sends it a randomly selected outgoing server.

This Subchain transaction has the receiver's address and the amount of NAV to send encrypted and attached to it. When the outgoing server receives the Subchain transaction, it decrypts the data, randomizes the transaction amounts and sends the NAV to their intended recipient from a preloaded pool of NAV that is waiting on the outgoing server.

After the outgoing server has sent out the randomized NAV to the intended recipient, the incoming server will join together any NAV which has been processed and on the next transaction cycle send it to the outgoing server to replenish the preloaded pool of NAV for future transactions.

The consequence of this is that we have broken the transactional link between sender and receiver on the Nav Coin blockchain by routing the transaction information through the Subchain. The NAV sent to the recipient are not in any way connected to the NAV that are received.

We can reliably instruct the outgoing server to send NAV to the recipient before the sent NAV arrive because we can trust the information provided by the Subchain as accurate due to the immutable nature of blockchain transactions.

SOLUTION DETAILS

TECHNOLOGIES USED

Nav Coin and the Subchain are both built using C++ and QT Creator. This is the language used to create most cryptocurrencies available today.

The Navtech processing servers are built using Javascript. They use NodeJS and Express to run a simple web server which allows for HTTPS communication between the wallet and the servers. They are written using ES6 Javascript which is cross compiled with Babel at runtime.

The system is tested running NodeJS v6.6.0 which is the current version at the time of writing.

Navtech is run on Ubuntu 14.04 to allow for the simplest compiling environment for the Nav Coin and Subchain daemons.

RSA key pairs are used to encrypt all data committed to the blockchain. RSA key pairs are asymmetric in nature which allows us to broadcast the public key to the user's wallet without worrying they could use that key to decrypt someone else's encrypted data.

It is also recommended to use git command line for keeping up to date with the latest revisions of the Navtech, Nav Coin and Subchain repositories.

SYSTEM OVERVIEW

Navtech servers operate in trusted clusters. One server acts as a recipient for transactions sent from a user's wallet. Another acts as the sender to dispatch transactions to the intended recipient. They will be referred to here as incoming and outgoing servers.

Both the incoming and outgoing server will be required to each run the Nav Coin (navcoind) and Subchain (subchaind) daemons so it can read and create transactions on both blockchains. Full instructions for installing and compiling each of these daemons is available in the readme associated with each GitHub repository.

Once each server is able to run navcoind and subchaind the Navtech project files will need to be downloaded onto each server also. The dependencies for the Navtech project are currently installed with Node Package Manager and full installation and setup instructions are available in the readme of its GitHub repository.

SOLUTION DETAILS

NAVTECH SETUP

Each server instance has various settings files. They configure static variables, user settings and settings specific to each type of server. Configuring them is outlined by the readme.

Before processing transactions, each server needs to be initialized using an npm command which runs the setup procedure. Firstly the setup procedure will check the settings files the user has configured and make sure they have correctly filled the required fields, and they match the minimum validation requirements.

The setup procedure then checks if it can communicate with and unlock navcoind and subchaind using the credentials provided by the settings files. If either daemon is found not to be encrypted, it will encrypt each wallet with the wallet pass phrases provided by the settings files ensuring the server runs using password encrypted wallets.

Once the provided credentials can decrypt both daemons the procedure will generate the server's initial RSA private and public key pair. The key pair will be tested by encrypting a static wallet address with the public key and then decrypting the output with the private key.

After the keys are created and tested, the procedure will create a pool of Nav Coin and Subchain addresses to be used for sending and receiving transactions. The standard mode is to generate a set amount of addresses to use, as defined by the settings files. There is some provision to use a new address each time a transaction is created but as yet the issue of wallet bloat over time is unaddressed at present so this option is disabled.

When the address counts for each daemon reach the maximum amount specified in the settings files, the procedure will generate a 42 character salted secret which is used as a private secret known only to each server in the trusted cluster. This private secret needs to be copied into the settings file for all the incoming and outgoing servers and is used in a security layer as outlined later in this document.

Both the incoming and outgoing servers are configured in the same way, except the outgoing server will not generate a private secret due to the fact they need to match on each server. The secret used at runtime does not have to be the one generated, but it does need to be 42 characters in length and match on each server in the cluster.

SOLUTION DETAILS

WALLET CONFIG

When a user wants to send NAV anonymously, the will need to specify which anonymous servers they want to use in their navcoin.conf file. The parameter addanonserver accepts a string which consists of an IP address and a port number in the IPV4 format 192.168.0.1:3000. If no port is specified it will default to using the port 443 which is standard for HTTPS requests.

The user also can add temporary anon servers by using the RPC command console and entering the command there “addanonserver 192.168.0.1:3000 add”. There are also list and remove commands available here. Servers added this way are non-permanent and will be lost when the wallet is shut down.

NAVTECH API

Navtech doesn’t rely on a central API as this can be a weak point. Each server runs its own local API instance which can perform various necessary tasks for the systems operation.

The main routines the API runs are as follows:

- The Nav Coin wallet queries a random incoming server’s API to check that the selected incoming server is online and able to process transactions.
- The incoming server confirms it was able to decrypt the wallet address encrypted by the Nav Coin Wallet before the Nav Coin transaction is committed.
- The incoming server queries a random outgoing server’s API to check that the selected outgoing server is online and able to process transactions.
- The outgoing server confirms it was able to decrypt the wallet address encrypted by the incoming server before the Subchain transaction is committed.

The specifics of each of these routines will be elaborated on at the appropriate parts of this document which relate to that part of the process.

SOLUTION DETAILS

CREATING THE WALLET TRANSACTION

When the user checks the “Send NAV Anonymously” checkbox in the Nav Coin QT wallet, the wallet retrieves the anon servers added to both the navcoin.conf file and entered through the RPC console commands.

It will pick a random server from the list and make an HTTPS request to its API endpoint to determine if the server is online and able to process transactions.

If the server returns an inadequate response or is not contactable, the wallet will pick another server at random and continue to attempt to find a valid server until it finds one or runs out of servers to try.

To determine if a server can transact, the server must pass the following checks:

- Unlock both navcoind and subchaind with the wallet passphrase specified in the settings.
- Retrieve the latest RSA key pair from the file system or generate a new pair if the key pair is considered out of date.
- Successfully encrypt and decrypt data using the retrieved key pair.
- Select a random NAV address from the address pool generated on setup.

If these checks pass, the incoming server will respond to the Nav Coin wallet with its randomly selected Nav Coin address and the incoming server’s public key. The RSA keys used by the servers are short lived and regularly deleted.

The wallet then encrypts the receiver’s address with the provided public key. It checks whether the encrypted string length is correct and if there was some problem encrypting the address it will try again up to a maximum of 10 retries.

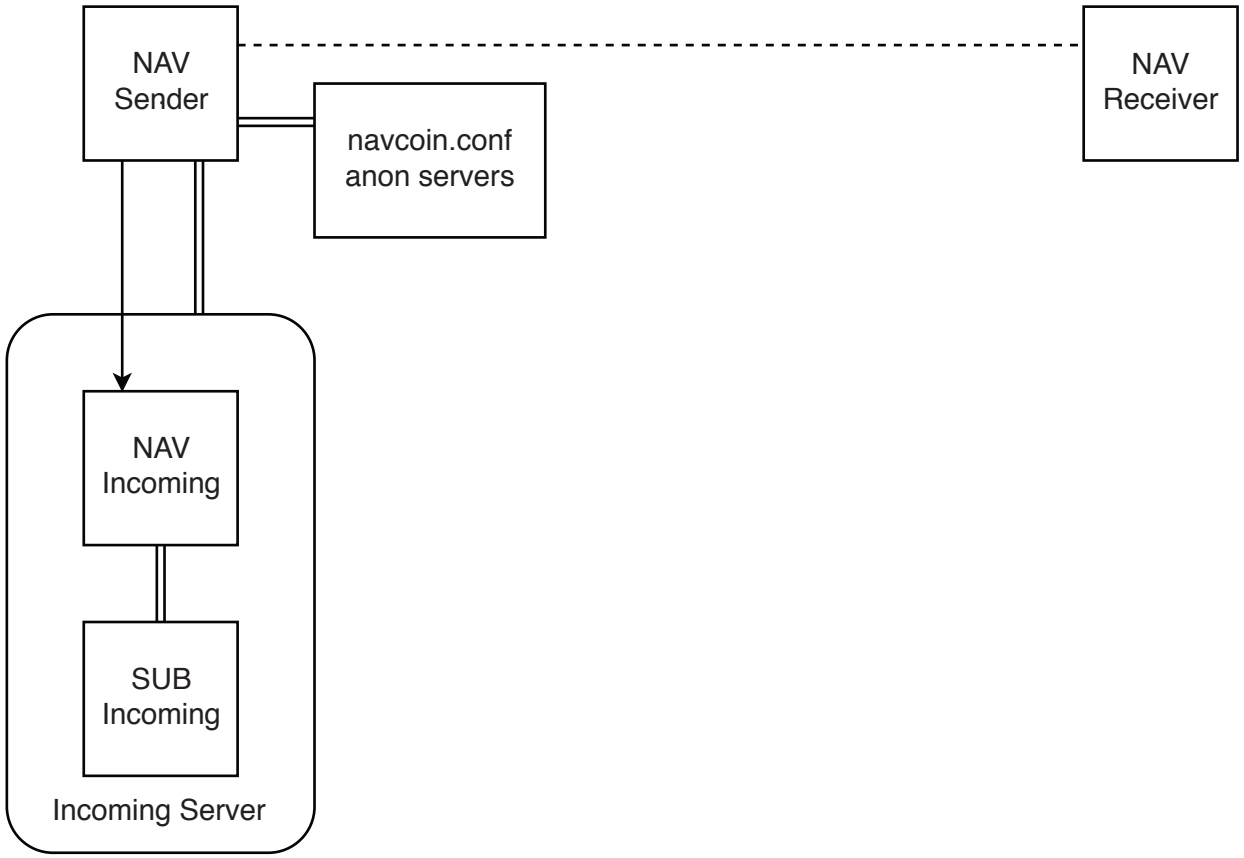
Once the wallet assumes it has successfully encrypted the receiver's address it will send it to the incoming server which it received the public key from to make sure the server can decrypt the address before it commits the transaction to the blockchain.

SOLUTION DETAILS

If the incoming server can decrypt the address, it will return a positive response, and the wallet will proceed to commit the transaction. Instead of sending it to the receiver's address, it sends the chosen amount of NAV to the random NAV address provided by the selected incoming server and attaches the encrypted receiver's address to the additional field "anon-destination" which we have added to our blockchain schema.

The transaction amount must be between 10 and 10,000 NAV. We have restricted it as such so that we don't require having unnecessarily large pools of NAV waiting on the outgoing servers to fulfill transactions.

Figure 1.1 Creating the Wallet Transaction



Instead of sending NAV directly to the receiver, the wallet encrypts the receiver's address and sends the transaction to one of the addresses provided by the randomly selected processing server.

SOLUTION DETAILS

RECEIVING AND PROCESSING THE NAV ON THE INCOMING SERVER

Every 2 minutes a procedure runs on the incoming server to process any NAV transactions which it has received and have reached a minimum of 1 confirmation on the network.

Firstly the script checks to see if the previous cycle has successfully completed. If it has not yet finished processing, it will skip the transaction cycle so it does not accidently process the same transaction twice.

The next check is to see if navcoind and subchaind are fully synced. It will not process transactions if either wallets are not fully synced with the latest block as reported by getblockcount. The daemons sometimes time out when they are attempting to catch up to the latest block because they are busy running the routines required to check the blocks it is catching up on. Therefore it has been made mandatory to be up to date with both blockchains before attempting to process transactions.

Then the incoming server will attempt to unlock both navcoind and subchaind with the provided wallet passphrases in the settings files. If it can not unlock either wallet it will not attempt to process the transactions.

The incoming server will then choose a random outgoing server from the trusted cluster and communicate over HTTPS with the outgoing server's API to get the current balance of its NAV pool. Once it knows how many NAV the outgoing server will be able to forward, it can begin to build the list of transactions which it will process this cycle.

The outgoing servers only accept API requests from the whitelisted incoming server IP addresses which are configured to be part of the trusted cluster. IP whitelisting and the use of a shared secret key are the main security features which prevent unauthorised transactions being processed by an outgoing server once the system is decentralized.

The incoming server retrieves all unspent transactions from its navcoind wallet. It then retrieves all navcoind wallet addresses which are created against the account name which we assigned to the pool of addresses created during setup. It then intersects these data sets to filter out any unspent transactions which may have been sent to the navcoind wallet but which are not intended to be processed as anonymous transactions.

SOLUTION DETAILS

If there are no intersections, the procedure will exit and report to the terminal that there is nothing to process.

If transactions are waiting to be processed the server will then prune this data set so the current batch of NAV we attempt to process adds up to less than the total number of NAV which are available in the outgoing server's NAV pool. This is so we can ensure the transactions we process are able to be forwarded immediately when they reach the outgoing server.

While pruning the list, it makes sure that the number of individual transactions are less than the total number of addresses in the outgoing servers Subchain address pool. This is to ensure that each transaction made during one processing cycle is sent to a unique Subchain address.

It also prunes transactions from the list if it doesn't have enough subchain coins to create the necessary transactions to the outgoing server.

Once this pruning is done, we are left with what we will call the current batch of transactions which we will process during this transaction cycle.

Now that we know the size of the current batch, the incoming server will request the needed number of SUB addresses from the outgoing server and the other details it requires to transact. To be considered able to transact the outgoing server must pass the following checks:

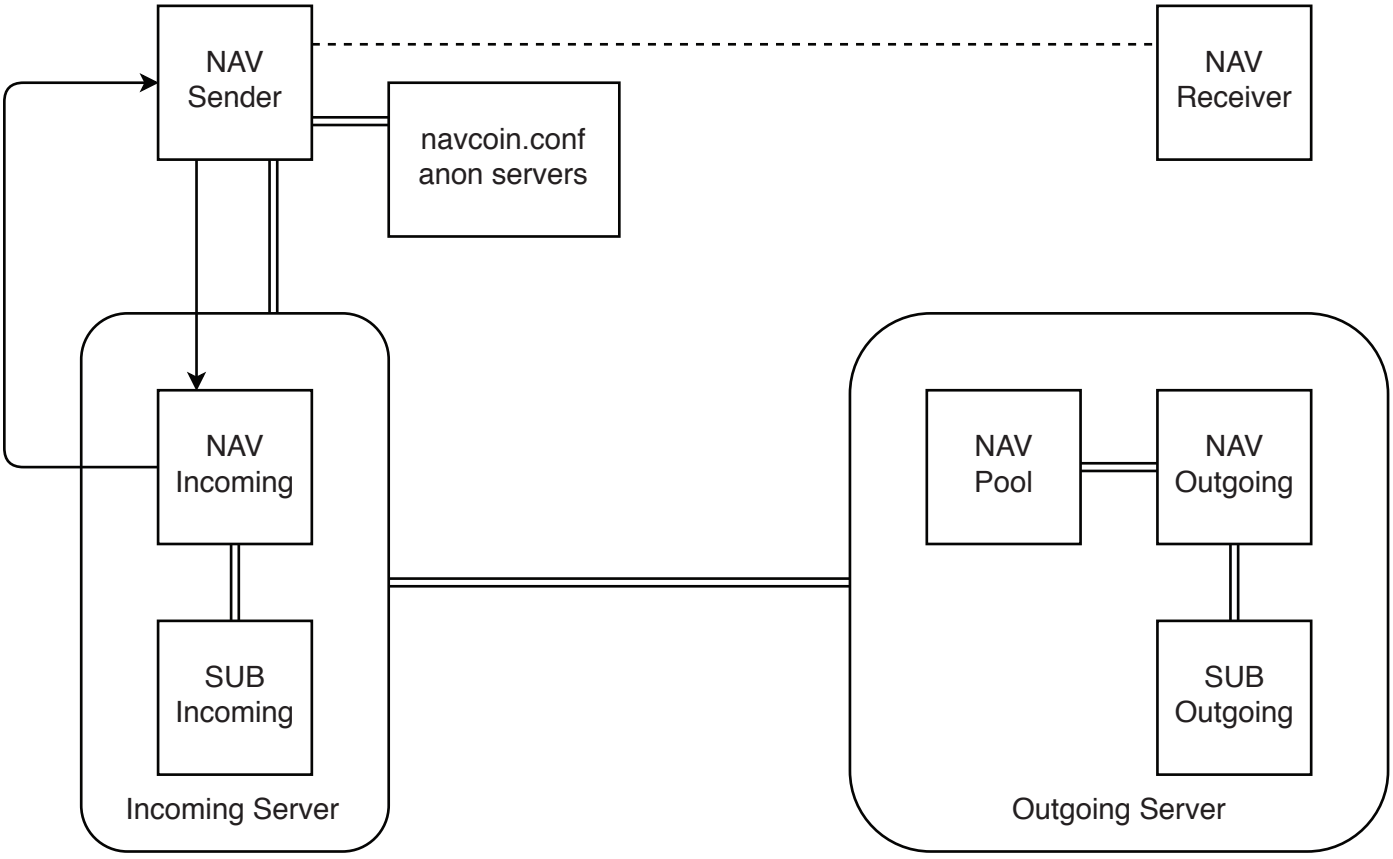
- Unlock both navcoind and subchaind with the wallet passphrase specified in the settings.
- Retrieve the latest RSA key pair from the file system or generate a new pair if the key pair is considered out of date.
- Successfully encrypt and decrypt data using the retrieved key pair.
- Select the requested amount of random SUB addresses from the address pool generated on setup.

SOLUTION DETAILS

If these checks pass, the outgoing server will respond to the incoming server with its randomly selected requested amount of SUB addresses and the outgoing server's public key.

The incoming server users the outgoing servers public key to encrypt some test data. It then checks the returned SUB addresses are valid wallet addresses. If all these checks pass, the incoming server will begin to process the current batch. If something goes wrong in these steps, the entire current batch will be returned to sender.

Figure 1.2 Receiving and Processing the NAV on the Incoming Server



Once the incoming server receives a transaction it will contact a randomly selected outgoing server and see if it can transact. If no servers can transact it will return the NAV to the sender.

SOLUTION DETAILS

SENDING THE SUB TO THE OUTGOING SEVER

Once the incoming server has determined the current batch and confirmed the outgoing server has provided enough information to proceed, it will begin to process the current batch. The first transaction to be chosen for processing will be the oldest transaction in the list.

The incoming server will attempt to decrypt the receiver's address before continuing. It will look up all private keys in the designated directory and attempt to decrypt with each of them until either a successful decryption has been detected or all the keys have been tried. If the server was unable to decrypt the receiver's address, it will return the NAV to the sender.

If it was able to decrypt the data, it will then check the decrypted data is a valid NAV address. If the address can be validated, the server then constructs the data which it will encrypt and attach to the SUB transaction. This time, we encrypt not just the address but also some additional information:

- address: receivers NAV address.
- amount: amount of NAV the sender sent to the receiver.
- secret: the shared private secret known only to the trusted cluster.

The server will then encrypt this data with the public key which was provided by the outgoing server. Again, he RSA keys used by the servers are short lived and regularly deleted. Once encrypted, the length of the output is checked and if it is incorrect, the encryption will be attempted up to ten times before being rejected, and the NAV returned to sender.

If the data is correctly encrypted, the server then proceeds to create the Subchain transaction of arbitrary size to one of the addresses provided by the outgoing server and add the related NAV transaction to a list of successfully processed transactions.

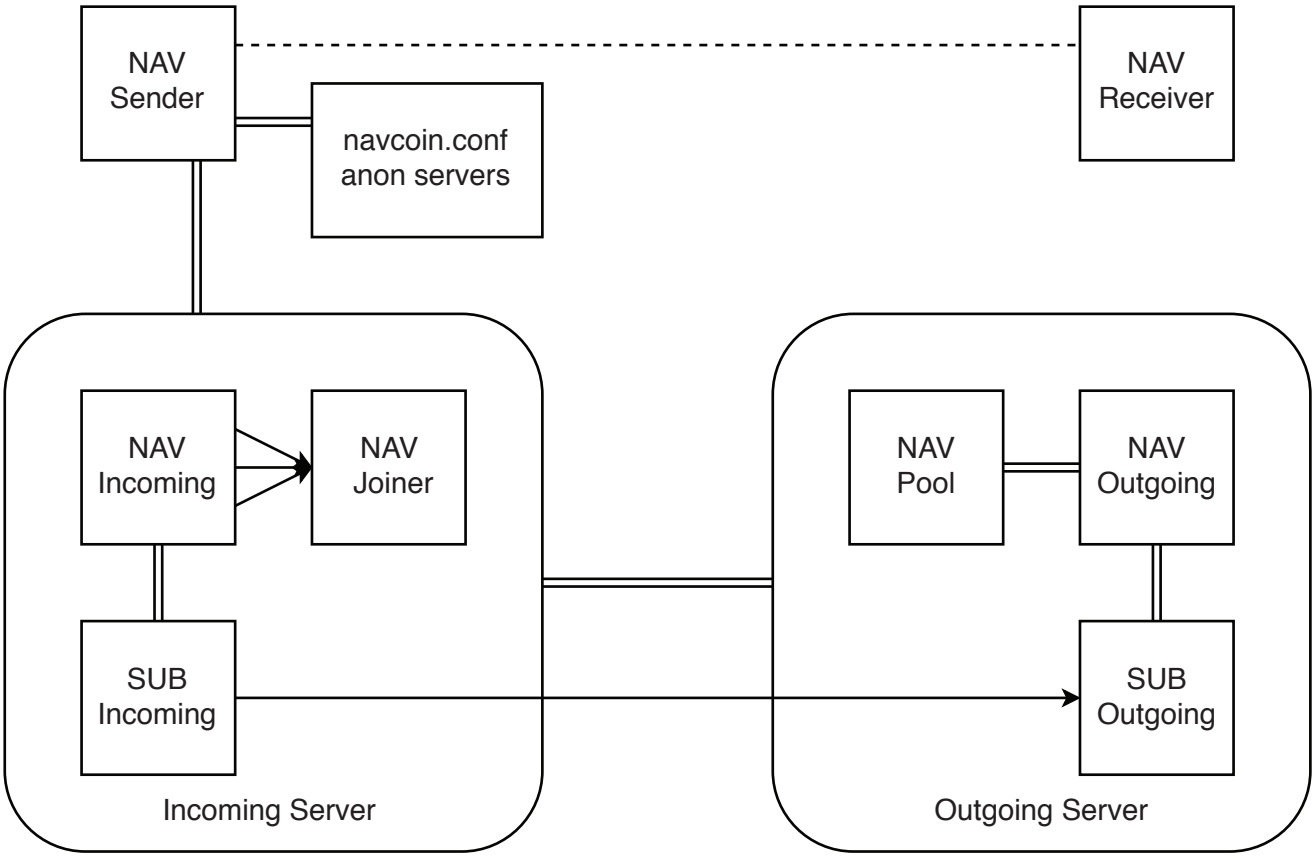
Once all transactions in the current batch have been processed and the corresponding SUB transactions committed to the Subchain, the server will sum the NAV balances of the successfully processed transactions and prepare to spend those amounts to remove them from the list of unspent transactions.

SOLUTION DETAILS

These NAV transactions are grouped together and sent to a randomly selected holding address owned by the incoming server.

When all the necessary SUB transactions have been created and all processed NAV has been sent to the holding address, the incoming server is released and considered available for processing the next batch.

Figure 1.3 Sending the SUB to the Outgoing Server



The incoming server re-encrypts the intended address, amount and private secret, then sends a SUB transaction with the attached information to a random address provided by the outgoing server. It also joins all the NAV processed during this transaction cycle together into a random address provided by the Incoming Server.

SOLUTION DETAILS

RECEIVING AND PROCESSING THE SUB ON THE OUTGOING SERVER

The process here is much the same as when NAV is received at the incoming server but with some key differences.

Every 2 minutes a procedure runs on the outgoing server to process any SUB transactions which it has received and have reached a minimum of 1 confirmation on the network.

The script checks to see if the previous cycle has successfully completed. If it has not yet finished processing, it will skip the transaction cycle.

The next check is to see if navcoind and subchaind are fully synced. It will not process transactions if either wallets are not fully synced with the latest block as reported by getblockcount.

Then the outgoing server will attempt to unlock both navcoind and subchaind with the provided wallet passphrases in the settings files. If it can not unlock either wallet it will not attempt to process the transactions.

The outgoing server retrieves all unspent transactions from its subchaind wallet. It then retrieves all subchaind wallet addresses which are created against the account name which we assigned to the pool of addresses created during setup. It then intersects these data sets to filter out any unspent transactions which may have been sent to the subchaind wallet but which are not intended to be processed as anonymous transactions.

If there are no intersections, the procedure will exit and report to the terminal that there is nothing to process. If transactions are waiting to be processed the server immediately begins to process them.

SENDING THE NAV TO THE RECEIVER

For each pending SUB transaction, the outgoing server will attempt to decrypt the attached data before continuing. It will look up all private keys in the designated directory and attempt to decrypt with each of them until either a successful decryption has been detected or all the keys have been tried. If the server was unable to decrypt the receiver's address it returns the SUB to the sender and logs the error to the server.

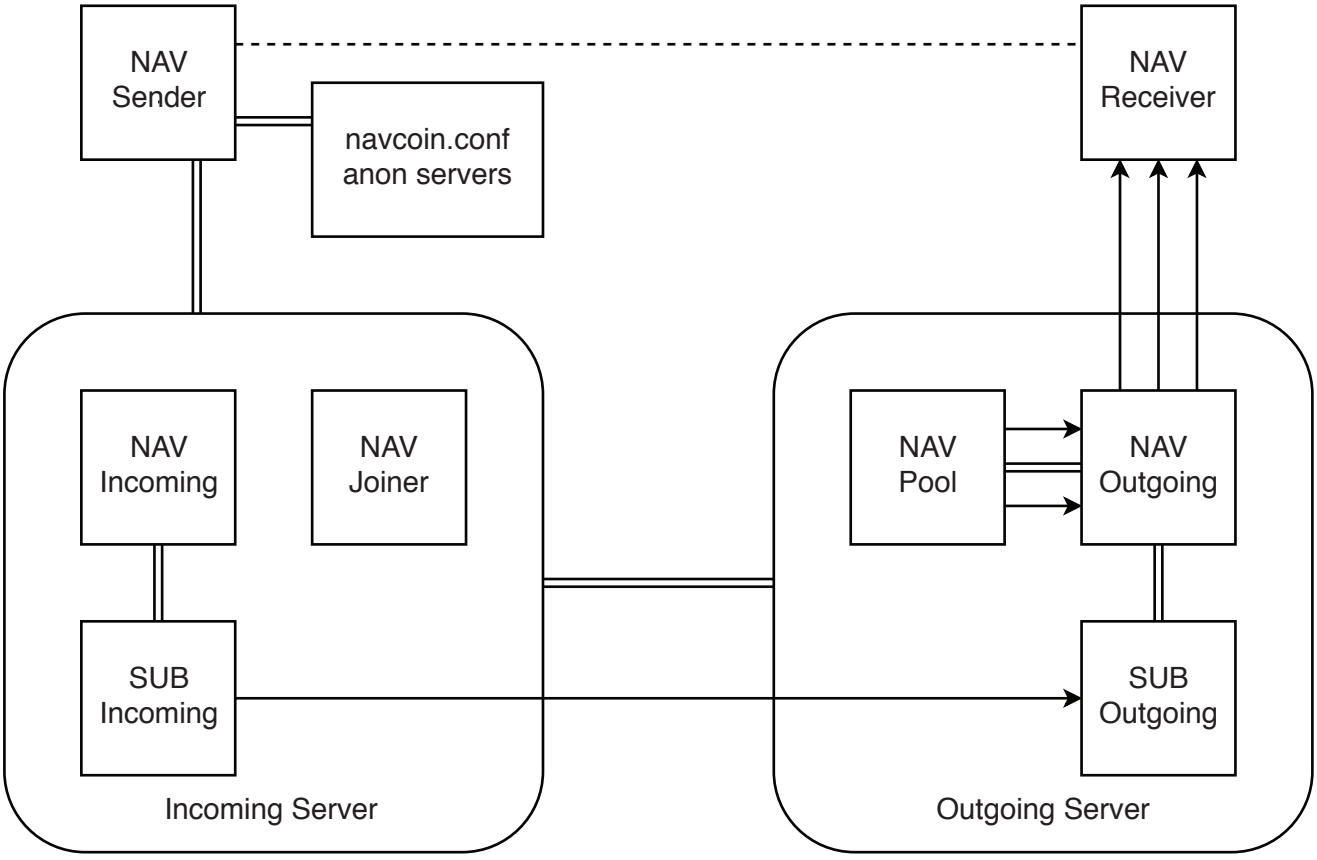
SOLUTION DETAILS

If the data was successfully decrypted but the amount or address invalid, or the decrypted secret doesn't match the private shared secret then the server will return the SUB to sender and log the error to the server.

As it cycles through decrypting the attached data, the outgoing server will sum the amounts and only add it to the the current batch if there are enough NAV in the pool to send out during this transaction cycle.

Once the current batch is determined, the outgoing server begins to process the current batch. It will randomize the decrypted amount into a random number of randomly valued transactions and send these to the decrypted receiver's NAV address from the pre-loaded NAV Pool.

Figure 1.4 Sending the NAV to the Receiver



When the Outgoing Server receives the SUB transaction, it decrypts the receivers address and sends the random NAV transactions from the NAV Pool.

SOLUTION DETAILS

RETURNING THE SUB AND REPLENISHING THE NAV POOL

If the NAV is successfully sent, the corresponding SUB transaction will be added to a list of successfully processed transactions.

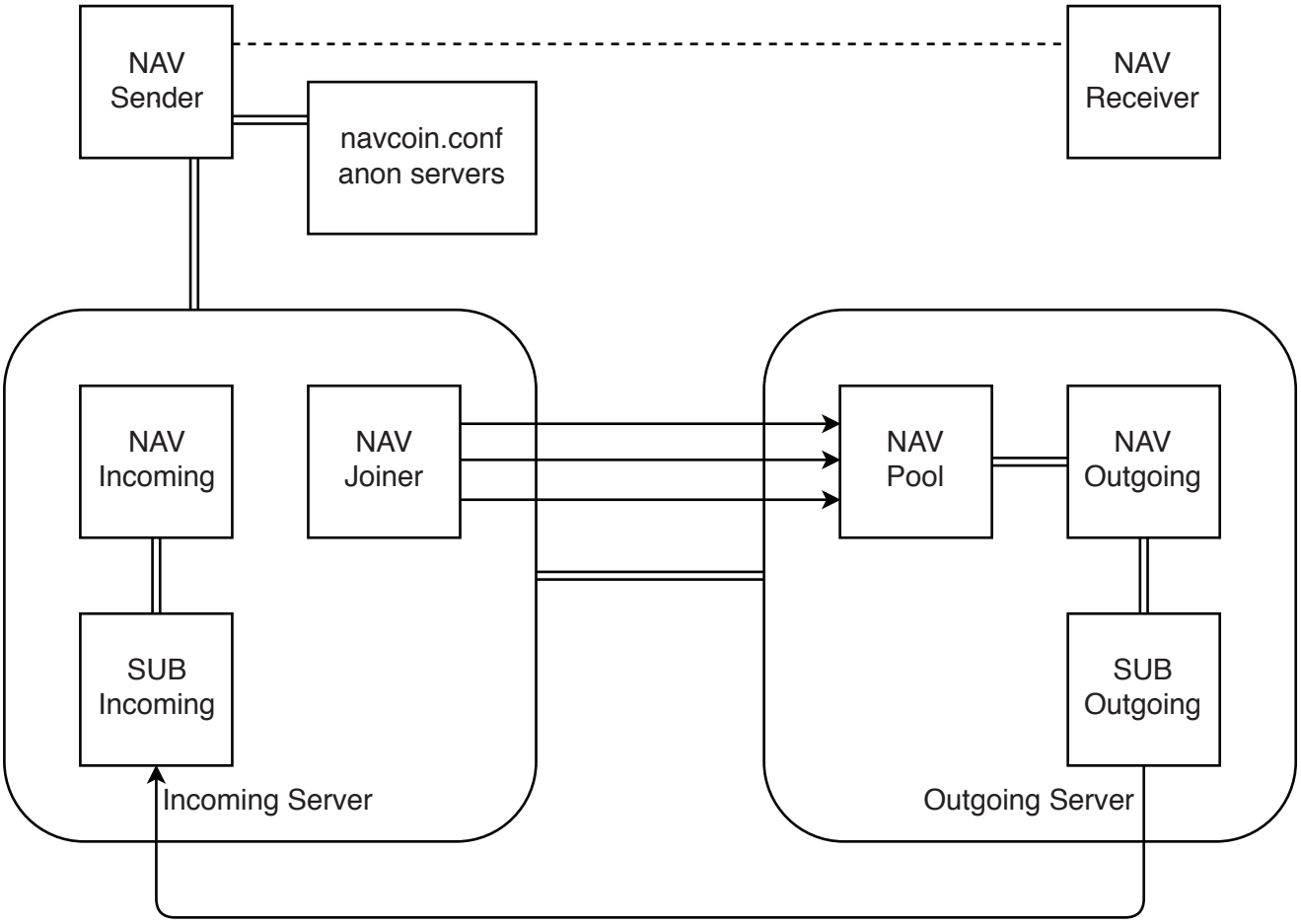
Once all transactions in the current batch have been processed the outgoing server will return the SUB to the incoming server, spending those amounts which then removes them from the list of unspent transactions.

When all the necessary NAV transactions have been created and all processed SUB have been sent to the originating Incoming Server, the outgoing server is released and considered available for processing the next batch.

After the NAV have been sent from the Outgoing Server, the balance of the holding address on the Incoming Server is split into a random number of transactions of random values and sent to randomly selected addresses owned by the outgoing server to replenish the Nav Pool to fulfil future transactions.

SOLUTION DETAILS

Figure 1.4 Returning the SUB and Replenishing the NAV Pool



The SUB are returned to the incoming server they originated from and the NAV are sent from the to the Outgoing Server which requires refilling.

SOLUTION DETAILS

ERROR HANDLING

This process is complex and there are a lot of moving parts. Most of the errors are handled in a preemptive manner. Meaning that both the incoming and outgoing server simply will not process a transaction or even an entire batch without meeting all of the necessary requirements.

If the incoming server runs into an issue mid-procedure and it cannot continue, it will simply either return the NAV to the sender or attempt to process the transaction again in the next transaction cycle.

Things get a little more difficult if the outgoing server runs into a mid procedure fault because it has no knowledge of the original NAV sender and can not return the coins. It is expected at this point due to our previous error checking that the transactions will be processed or it will attempt to process them again in the next transaction cycle. The worst case failover is to log the error to the server for post analysis and manual return to the sender upon investigation.

No identifiable information is ever stored on our servers.

TECHNICAL BENEFITS

Using a Subchain as the transaction director between the incoming and outgoing server gives our system multiple competitive advantages over using legacy architecture. The Subchain is based on the same underlying technology as the main Nav Coin blockchain and therefore possesses all the advantages which come along with blockchain technology.

The blockchain of both Nav Coin and the Subchain are decentralized, offering unparalleled resilience and restorative capacity. The distributed blockchain enables a server to be restored from the wallet backup generated on setup with no need for any data to be routinely backed up.

If a server was recovered from the wallet back up, it would simply begin processing from the oldest unspent transaction listed as unspent in its addresses. In a lot of recovery cases, this would cause the transactions to simply be returned to the sender if the encryption keys were lost. If the keys are intact the server will continue to process from where it left off.

The other benefit of routing transactions through the Subchain is that it breaks the connection between the sender and the receiver on the Nav Coin blockchain. The amount of Subchain coins sent per transaction is flat and is in no way related to the amount of NAV sent. The data attached to the Subchain transaction is also encrypted with a different encryption key and it is impossible to link Subchain transactions to their corresponding NAV transactions by analysing the blockchains.

Because blockchains are trustless (as in there is no need to trust a third party because the rules are impossible to break), we can rely on the information encrypted into the SUB transaction received by the outgoing server is valid. This allows us to send NAV to the receiver from the preloaded NAV pool before any NAV from the original transaction even reach the outgoing server. They are not able to be transactionally linked in a linear timeframe, and this guarantees the NAV received are not the NAV sent and can not be traced to source through the Nav Coin blockchain.

Both blockchains operate with a 30 second block time which means transaction confirmation occurs quickly, and transactions are processed rapidly through the system. A typical timeframe between sender and receiver is 5 minutes.

With setup only requiring a few simple steps and the trust generated by using a blockchain based system, means that we can open the system for other people to set up their own processing server clusters in the future.

BUSINESS BENEFITS AND FUTURE GROWTH

All of the technical benefits add up to a system which is fast, redundant (as in easily restorable), resilient, trustless and completely anonymous. This will lead to higher user confidence and wider adoption.

The system accrues transaction processing fees which are a low percentage, but given enough volume will become a substantial revenue stream for operators.

Because this system is designed to be distributable, when we decentralize it, other entities will be able to set up their own servers and earn revenue by processing transactions.

The decoupled design of the system also allows for the potential addition of multiple cryptocurrencies which all send anonymous transactions through the Subchain.

The distributed nature of the blockchain technology used means the infrastructure required to run the system is minimal, reducing ongoing overheads.

SUMMARY

KEY BENEFITS

Navtech is fast, simple to use and totally anonymous. Users can have high confidence in a secure system which has proven blockchain technology as its information backbone.

We have analysed the market and worked hard to provide a solution which solves what we see as the major problems faced by financial privacy advocates in the modern era.

The technology we have used minimises the risk of lost transactions, even if the processing servers were literally wiped from the face of the planet mid-processing cycle.

The encryption we use is so strong that it would take a supercomputer more than 100 billion years to decrypt the data we commit to the blockchain.

The security protocols we employ are impossible to circumvent, and the system is unable to send anything other than legitimate transactions.

RISKS

As with all blockchains, there are a small number of associated risks when dealing with democratically agreed distributed data. The main risk being a 51% attack. Currently we are the only ones with access to the Subchain coins and since it is Proof of Stake a 51% attack by an outside entity is impossible.

The other risk factor is server security of the processing servers themselves. We have done everything in our power to protect the servers from being compromised and will monitor and evaluate our procedures as time passes.

SUMMARY

CONCLUSION

Nav Coin's Navtech system is a world class anonymous transaction suite and the first of its kind. It uses an innovation in blockchain technology to solve the inherent problems of using a public ledger while staying true to the principles cryptocurrency.

http://navcoin.org

http://twitter.com/NAVCoin

http://facebook.com/NAVCoin

http://reddit.com/r/NAVCoin