

FLASH Whitepaper

Crypto coin technology to power online social and media applications.



Abstract

FLASH is a blockchain based platform that enables users and developers to leverage this powerful technology for social media, websites, blogs and ecommerce sites. It is limited only by the user's imagination but at the same time it creates an inherently level playing field where each contribution is fairly valued by the community in a completely transparent fashion.

September 1, 2016

Contents

Abstract
Table of Contents
Introduction
Legal Compliance
How to Contribute and Give
FLASH Architecture Overview
FLASH Account Structure
FLASH Key Generation, Storage and Recovery
FLASH Coin Supply
FLASH Coin Distribution
Security of Unissued Coins
FLASH Blockchain
Scalability and Performance
FLASH Cloud Wallet
Wallet Component Overview
Exchange (future)
IT Infrastructure
Appendix: Wallet Webservice API

Introduction

The fundamental principle of FLASH is that all work and/or contribution to the network should be valued by the community in an objective fashion. Allowing the free market process to function creates a mechanism whereby all forms of work can be reduced to a common denominator FLASH.

What this means is that any form of work, be it valuable time, a user's work and attention, a special skill set such as developing tools, various forms of energy (ie processing) and currency can be valued in real time based on the market supply and demand for that specific work/contribution.

The ways to contribute to the community are only limited by one's imagination and the community's collective valuation of it. At its core FLASH is simple to understand, completely transparent with no complicated rules and conditions that are hard to work with. Because FLASH has a very simple and fast settlement system (< 2sec!), and built in scalability (~ 25,000 transactions per second), it will have many use cases.

Legal Compliance



FLASH was designed from the ground up to be legally compliant through each phase of its life. While the FLASH coins are being given away free today and they have no redemption value whatsoever, it's possible that could change in the future. To safeguard our users, there are policies and procedures that we are putting into place today to ensure everything is done lawfully and legally.

Our systems include the following:

- A way to store, protect and recover customer keys
- An easy to use secure wallet
- Monitoring of transactions to prevent abuse
- A Know Your Customer program for transactions over a certain amount
- An Anti-Money Laundering program
- Reporting suspicious transactions and maintaining records of all transactions (using the blockchain)
- The ability to geo-fence out certain countries by IP (New York. is currently not supported)

There are two types of accounts with Flash, basic and validated. The characteristics and limits of each account are:

BASIC

Validated email address
No New York users allowed
No ID or KYC required
Daily Limit: \$3,000CAD per day
Wallet Limit: \$9,500CAD per account

VALIDATED

Validated email address
No New York users allowed
Govt issued ID required
Address, city, state, postal code, country, mobile
Daily Limit: \$9,500CAD per day
Wallet Limit: \$50,000CAD per account

The limits may change based on company and Canadian government regulator policy changes. If the coins in a user's basic wallet exceed the maximum amount, we may ask for ID and KYC items from the user.

How to Contribute and Give

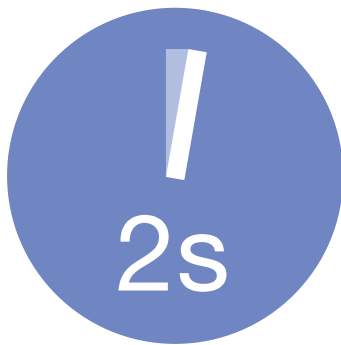
FLASH has been designed around the only successful and proven economic model -- the free market system. There have been many unsuccessful attempts in the past to partially replicate this or circumvent it entirely by creating artificial voting schemes and complicated ownership arrangements. FLASH is unique in that the market is the only arbiter of value which is free to fluctuate based purely on the supply and demand forces in the community. The fundamental unit of account on the FLASH platform is the FLASH cryptocurrency token. While we expect that there will be many different ways to contribute limited only by one's imagination here are a few use cases:

.....
In short, anything that requires end-to-end security, near real time settlement, certain conveyance of value or information, and a permanent audit trail are ideal for FLASH. When you develop an application with FLASH, you can build your fee or payment right into the application, too.
.....

- Sign up with your email (required for account)
- Optionally add and validate your cellphone
- Refer another user to get a FLASH wallet
- Fountain to dispense FLASH to visitors at a web page
- Like button for a visitor to reward a blog/website owner
- Advertising fountain for the user and to support writers paid by advertiser
- Paid for content
- Registration and validation of ownership of digital media or physical goods
- An email postage stamp to validate the sender and compensate the receiver

FLASH Architecture Overview

FLASH IS A PREMINE, PERMISSIONED BLOCKCHAIN, BASED ON THE ORIGINAL BITCOIN/LITECOIN BLOCKCHAIN. IT HAS BEEN OPTIMIZED FOR SEVERAL VERY SPECIFIC USE CASES:



1. High performance. Transactions settle in less than 2 seconds and capable today of 25,000 transactions per second.
2. Highly secure. A number of security flaws with the original bitcoin blockchain have been patched. The platform has been enhanced with Curve25519 Elliptic Curve encryption. All communications between wallet and blockchain are in secure EC tunnel with curve ZeroMQ (<http://curvezmq.org/>).
3. Commercial scale throughput capability. The FLASH platform has been successfully tested at 25,000 trans/sec. Based on our lab tests we are confident that the platform can be optimized to well over 100,000 trans/sec in the near future, if the need arises.
4. A key recovery system to prevent loss of keys.

In order to reach these critical metrics for wide scale use by millions of participants the FLASH platform is based on the principles of a permissioned blockchain. The platform leverages both the existing enterprise hardened distributed database technology with blockchain technology characteristics, decentralized control, immutability and creation and movement of digital assets. FLASH has pre-mined 900 billion coins which will be distributed over the course of two years.

The FLASH system builds upon three tier application platforms. Like most standard systems we have **User Interface**, **Communications**, and **Business Logic / Storage** tiers.

The **User Interface Tier** enables the end user or application to interact with the FLASH System. The FLASH platform leverages HTML5, CSS3 and JavaScript on the browsers, with no extensions, enabling seamless cross browser support. We have developed and adopted all technologies that enable JavaScript to do what C/C++ and Java are able to do. In addition, the FLASH System uses Twitter Bootstrap to provide a responsive web framework that works on any device.

The **Communication Tier** enables a secure tunnel between the User Interface Tier and the **Business Logic / Storage Tier** without the need for OpenSSL. The secure protocols used by the FLASH includes:

HTTP / Web Sockets using Curve ZeroMQ and Libsodium. Our work in the area of communication also means that we are an early adopter of new secure protocols that run over JavaScript using HTML5.

The Business Logic / Storage Tier enables all of the transaction flows, business logic, and networking systems to be stored and operated within this layer. This tier is responsible for executing proprietary algorithms to store in the distributed database, which power FLASH.

- The **Key Exchange Cluster** is a horizontal cluster of servers that provide key exchange or key lookup for every transaction on the FLASH System, like a directory. Due the nature of cryptographic key pairs on the Bitcoin transactional system, every transaction requires a public wallet address lookup. Key exchange needs to act as the public address registrar. The Key Cluster also allows currency exchange, messages, public escrow and other information flow among peer-to-peer wallets. The message engine has enabled logging and notification of all activities on the wallet communication using Sendgrid email notification. The Key Server uses NodeJS, ZeroMQ, Redis, and MySQL.

- The **FLASH Wallet Application** is a horizontal scaling server application that enables all FLASH Wallets to be used from different Web Browsers. The FLASH Wallet Application has most of the Bitcoin Wallet functionalities and key exchange functionalities. It's a virtual online wallet using the HTTP Protocol / Web Sockets.
- The **Web Service API Servers** preprocess, convert and index all the FLASH transactions from the Web Interface and send them to the blockchain. This significantly speeds up the transactions as each wallet doesn't need to sync all blocks from the blockchain of local server. All wallets can share the blockchain on the Blockchain API server. This improves the transaction performance significantly due to the reduced network latency for each wallet synchronization. Another very important aspect of the technology is the significant reduction in the possibility of "double spending". All blockchain transactions are indexed, preprocessed and validated through the Gateway Server in a manner similar to Bitcoin.
- The **FLASH Blockchain** is a fork of Litecoin technology. A number of significant modifications have been made. The settings have been changed in order to speed up the transactions to the blockchain. All the coins have been premined and the mining has been reset to the minimal degree of difficulty factor. The blockchain is secured behind firewalls in multiple secure data centers. Traditional consensus along with intrusion detection is used to prevent the blockchain from being taken over by an attacker. The purpose of the FLASH Blockchain is to replace the traditional transactional database with a network storage database and include an end-to-end security data structure.
- Transaction Fees are set to one FLASH coin per transaction, these fees may be raised or lowered in the future, depending upon demand.



FLASH Account Structure

ACCOUNTS IN FLASH ARE STORED IN A CENTRAL AUTHENTICATION SERVER (CAS).

→
Each CAS account has the following fields:

id: account ID.

email: account's email

role: used for authorization, e.g: USER or ADMIN

privateKey: EC crypto private key (encrypted by user's password)

publicKey: EC crypto public key.

sc1: user's share used in recovery procedure (encrypted by user's security answers).

sc2: server's share used in recovery procedure

sc3: administrator's share used in case the user lost sc1 and cannot recover himself.

In addition user's profiles are stored in Flashcoin Key Server. The information includes: display name, avatar, country... which varies from app to app.

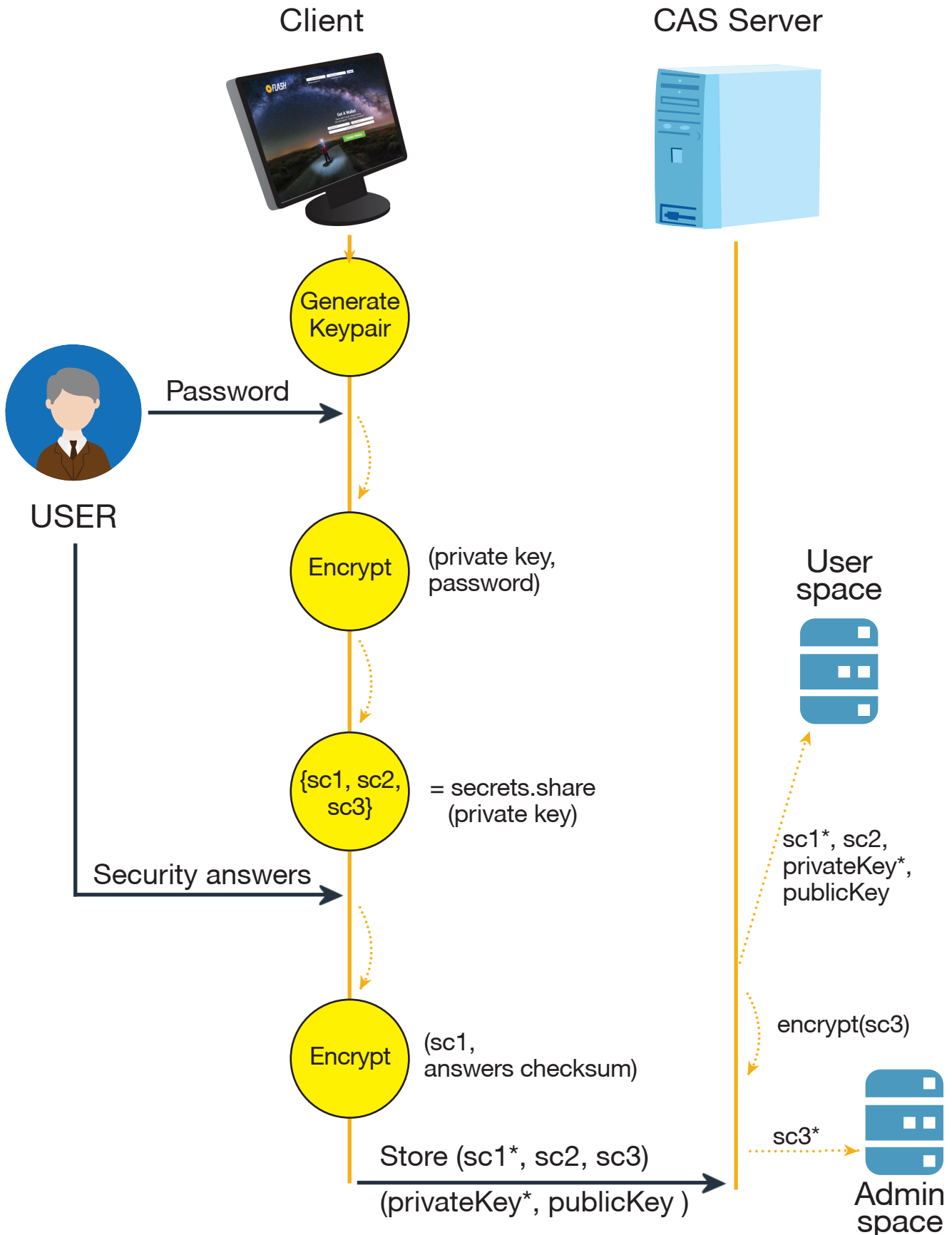
FLASH Key Generation, Storage and Recovery

Key generation at signup

EC crypto keypair is generated at client side when signing up. The private key is then encrypted by user's password. Recovery keys are also generated from the private key, which is a tuple (sc1, sc2, sc3). After user answers the security questions, the responses are then used to encrypt sc1.

Key storage

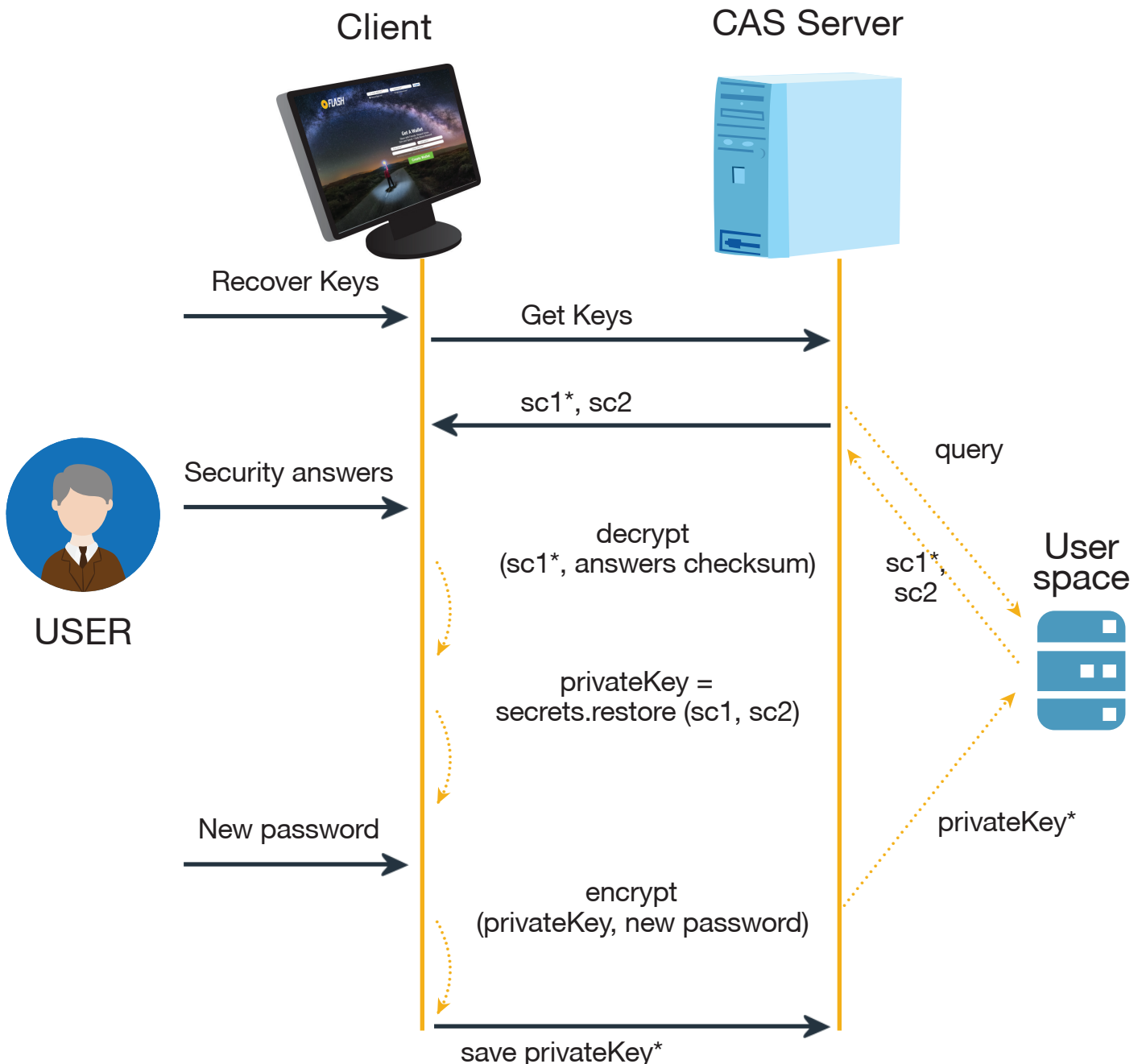
The server stores the following: the encrypted private key, public key, sc1 encrypted, security questions, sc2, sc3 (which is then encrypted separately by the admin).



Key Recovery

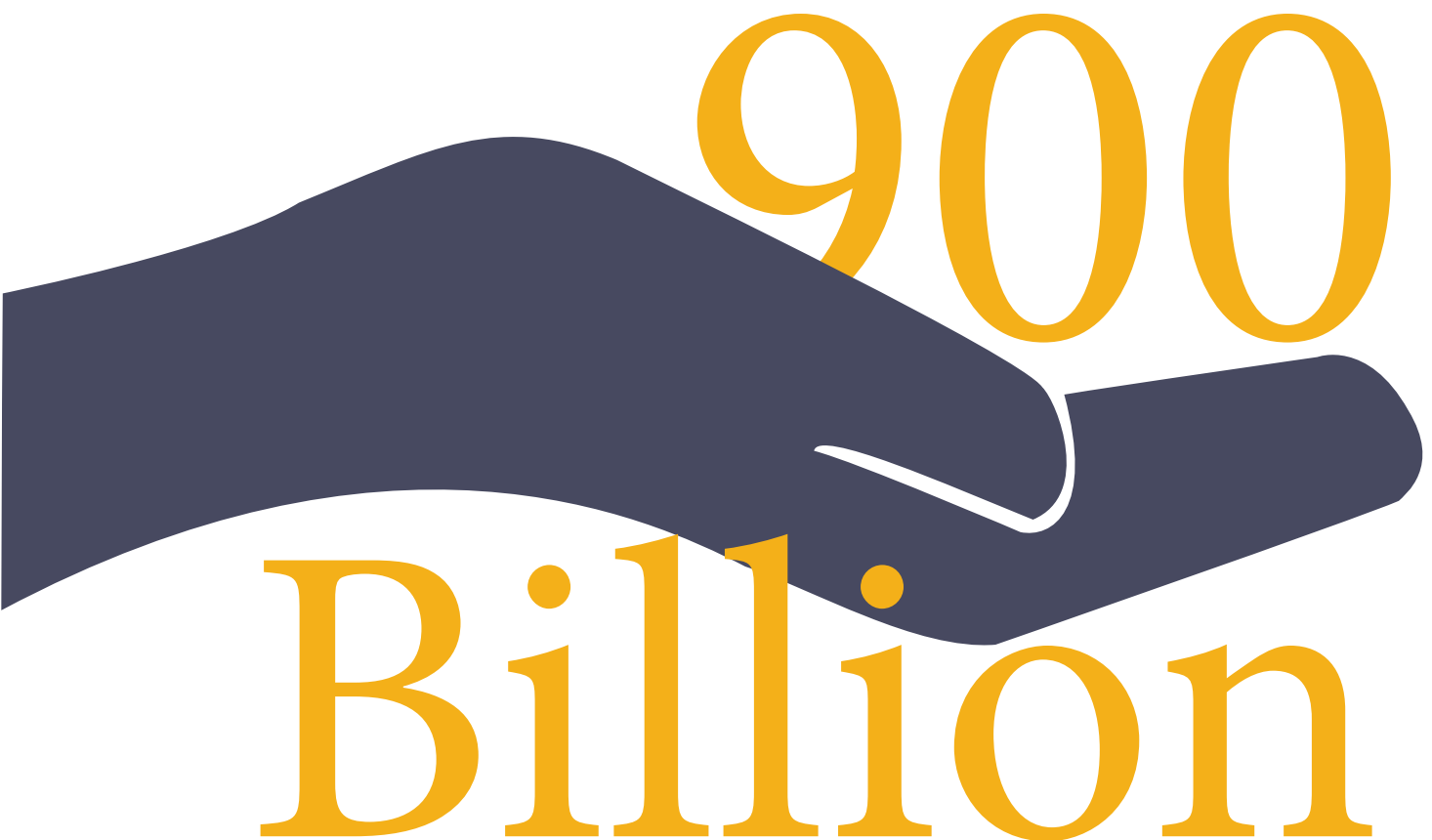
The recovery process is automatically triggered by the user. After the email is verified, the client receives sc1 encrypted and sc2, security questions. By answering the security questions correctly the answer is used to decrypt sc1. From sc1 and sc2, the private key is restored. User then needs to provide a new password and start the process of protecting and storing keys following the same as above.

User can also choose a super-secure mode where the server store an unique sc. When the recovery mode is activated, user must provide his/her share to combine with the server's share. If user lost the sc1 (given in the sign-up process) then no one can recover his/her password. Therefore, the compulsory participation of user in the recovery process ensures the security of user's share as well as the password.



FLASH Coin Supply

The FLASH coin supply is premined and consists of 900 billion coins. The amount was determined by modeling social media and blog usage and generating enough coins to distribute to millions of users, so they can have a non-fractional ownership of coins. Coins not issued are kept in cold storage (see Security of Unissued Coins below).

A stylized illustration of a hand holding the number 9000 Billion. The hand is dark blue with a white outline, and the number is large and yellow. The word 'Billion' is written in a large, yellow, serif font below the number.

9000
Billion

FLASH Coin Distribution

COINS WILL BE DISTRIBUTED IN FOUR TRANCHES.

1. Pre-sale of 5% of the total supply (45 billion) will be distributed to donors using Counterparty assets to help pay for the next phase of development of the Flash platform. (done)
2. Crowdsale of 10% of the total supply (90 billion) will be auctioned through a Bittrex administered crowd sale beginning October 25th. Please visit <https://bittrex.com/> for more details beginning in late October.
3. Finally, remaining FLASH coins will be given away to users, developers and others who help the FLASH network for free and sold to advertisers and business users, until the supply is exhausted.

—————→
**What happens
after the newly
issued coins
run out?**

People who would like coins for their projects or businesses will need to build something of value to induce other users to give them FLASH coin. Alternatively, in the future they could potentially purchase them from users or on an exchange at a market price.

Security of Unissued Coins

Unissued coins are stored offline in multiple remote locations. We will add multi-sig if the coin market cap requires it. Printed keys are kept and the drives are encrypted multiple times with unique keys generated using a hidden process. Only when new coins are needed are coins issued to general circulation and removed from cold storage.

FLASH Blockchain

FLASH HAS FORKED THE LITECOIN VERSION OF THE BLOCKCHAIN TO USE AS A DISTRIBUTED NETWORK STORAGE SYSTEM. A NUMBER OF SIGNIFICANT MODIFICATIONS TO THE CODE HAVE BEEN MADE IN ORDER TO ADDRESS THE WEAKNESSES OF FULLY DECENTRALIZED NETWORKS:

- **Network Latency** - Block synchronization among the nodes dramatically slows down the transaction validation (double spending) speed. FLASH is a closed network with guaranteed 1Gbs network pipe between the nodes. Because we use the blockchain as a transactional network database the network latency is guaranteed to be under 200ms (propagation time window).
- **Performance** - Two factors that determine the blockchain performance include Block Synchronization and Block Mining. FLASH uses cache and index servers to synchronize the nodes and reset the mining algorithm to the least difficulty factor. Because we have the trusted network of nodes, there is no need to continue to increase the degree of difficulty of mining for the block verification process. FLASH provides a unique solution to ensure distributed network storage data integrity.
- **Security Risk** - Block mining is vulnerable with open distributed blockchain network. FLASH Blockchain is not open to the public to mine or manipulate the blockchain by computational advantage.

End to End Encryption

In order to ensure no single point of weakness, the design of the security system and of all encryption functions have to be done from the wallet (client node).. As a result, transactions are encrypted by the recipient's public key which are then written into the FLASH blockchain. This methodology protects from intruders obtaining any encrypted data on the FLASH database. In order for any attacker or intruder to decrypt information, they must compromise the system and crack the Elliptic Curve

Cryptography (ECC) algorithms on each key. Even if someone had a quantum computer and was able to crack ECC the cost to decrypt a transaction would by far outweigh the possible gain. For an average computer it would take more than 100 Billion years in computation effort. Therefore, the cost of cracking the ECC on each transaction far exceeds the potential return.



Pre-Mined Tokens

We have mined about 900 billion coins which will be mainly distributed for free to users and developers

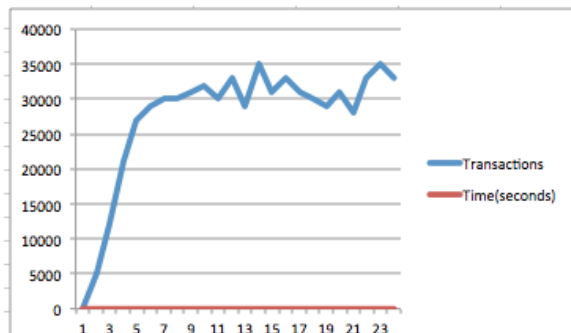
Blockchain API

A protocol that empowers Web Application to communicate with the FLASH BlockChain network. All transactions have been indexed at the Blockchain API layer to pre-compute and speed up transaction lookups such as double spending verification and transaction logs.

Mining

FLASH owns the mining pool for the entire FLASH Blockchain. For the purposes of scaling and redundancy of network data storage systems, FLASH uses enough CPU power/servers to setup a Network of Authority for all the Block Node Servers. Because we have reset the mining degree of difficulty to almost zero, the validation time per block is orders of magnitude faster than typical Litecoin or Bitcoin transaction processing.

Scalability and Performance



*Benchmark results for FLASH
running on single i7 processor Brix
cube at 3.9 Ghz with 16GB RAM
and 240GB SSD.*

The FLASH blockchain has been optimized for very high performance and scalability. Final settlement with blockchains is a concept based on the number of confirms by various nodes on the blockchain. A coin can't be reused until it reaches enough settlements to have very high confidence of the authenticity of the transaction and approval by the network. FLASH was optimized for final settlement time of under 2 seconds, including the ability to resend the coin.

Throughput is an important consideration, as a site or currency scales, it will reach "make or break" milestones. FLASH has been load tested and optimized to process 25,000 transactions per second and with minor modifications could easily reach 100,000-200,000 transactions per second. To put this in perspective, the VISA network peak load is about 50,000 trans/sec and Paypal's peak is in the low thousands of transactions per second.

APPENDIX

Wallet Webservice API

Create Account

Name: create_unverified_account

Description: Create unverified account (need to verify via email)

Request params: name, email, ip, callbackLink, g_recaptcha_response
(Google recaptcha response)

Response: {rc: Number}

Set Password and Verify Email

Name: set_password

Description:

Request params: password, privateKey (encrypted private key),
publicKey, token

Response: {rc: Number}

Get Session Token (sso)

Name: get_session_token

Description:

Request params: idToken, resource

Response: { rc: Number, profile : Object { sessionToken: String } }

Check Session Token (sso)

Name: check_session_token

Description:

Request params: sessionToken, resource

Response: {rc: Number, profile: Object{username: String, email: String} }

Login (sso)

Name:

Description:

Request params: email, password, ip, resource

Response:

Success: {rc: Number, profile: Object{email: String, display_name: String, gender: String, ...} }

Update Account

Name: update_account

Description: Update user profile

Request params: display_name, gender, profile_pic_url, about, timezone ...

Response: {rc: Number }

Get Profile

Name: get_profile

Description: Get user profile

Request params: {}

Response: {rc: Number, profile: {username: String, email: String, display_name: String, profile_pic_url: String ...} }

Set PIN

Name: set_pin

Description: Set PIN

Request params: pin

Response:

Success: {rc: Number}

Check PIN

Name: check_pin

Description: Check if PIN is correct

Request params: pin

Response:

Success: {rc: Number}

Change PIN

Name: change_pin

Description: Change the PIN

Request params: old_pin, new_pin

Response:

Success: {rc: Number}

Get Contact Details

Name: get_contact_detail_by_email

Description: Get contact details by email

Request params: contact_email

Response:

Success: {rc: Number, profile: {username: String, email: String, display_name: String, gender: String, profile_pic_url: String, ...} }

Get Profile

Name: get_profile

Description: Get user profile

Request params: {}

Response: {rc: Number, profile: {username: String, email: String, display_name: String, profile_pic_url: String ...} }

Get Users

Name: get_users_by_uid

Description: Get users information by user id

Request params: ['user1', 'user2', ...]

Response:

Success: {rc: Number, accounts: [account1, account2, ...] }

Get Roster

Name: ros_get

Description: Get contact list of a user

Request params: {}

Response:

Success: {rc: Number, roster: {total_subs: Number, subs: [], ...} }

Roster Operation

Name: ros_op

Description: operate roster, where operation could be REQUEST, APPROVE, REMOVE

Request params: op, from, to

Response:

Success: {rc: Number}

Notification: notify to related users

Create Wallet

Name: create_flash_wallet

Description: Create a new wallet

Request params: idToken, wallet_secret

Response:

Success: {rc: Number, wallet: {passphrase: String, wallet_id: String, address: String } }

Search Wallet

Name: search_wallet

Description: Search for wallet by keyword, to send money to

Request params: start, size, term

Response:

Success: {rc: Number, criteria, wallets: [wallet1, wallet2, ..], total_wallets: Number }

Get My Wallets

Name: get_my_wallets

Description: Get my wallets (currently only support 1 wallet)

Request params: {}

Response:

Success: {rc: Number, my_wallets: [], total_wallets: Number}

Add Transaction

Name: add_txn

Description: Push transaction to blockchain and add transaction log

Request params: receiver_id, amount, currency_type, receiver_public_address, transaction_id, memo, request_id, transaction_hex (signed)

Response:

Success: {rc: Number, id: String}

Notification: notify to the recipient about the new transaction

Get Transactions Log

Name: get_txns

Description: Get transaction log of current user

Request params: date_from, date_to, order, start, size

Response:

Success: {rc: Number, txns: [tx1, tx2, ...], total_txns: Number}

Get Transaction Log By Id

Name: get_transaction_by_id

Description: Get transaction detail by id

Request params: transaction_id

Response:

Success: {rc: Number, txn: {...} }

Create Unsigned Transaction

Name: create_unsigned_raw_txn

Description: Create a unsigned transaction to be signed by the owner later

Request params: from_address, to_address, amount

Response:

Success {rc: Number, transaction: {...} }

Get Transaction Details

Name: get_transaction_details

Description: Get transaction details from blockchain

Request params: transaction_id

Response:

Success: {rc: Number, transaction: {...} }

Get Balance

Name: get_balance

Description: Get wallet balance from blockchain api

Request params: {}

Response:

Success {rc: Number, balance: Number}

Add Money Request

Name: add_money_request

Description:

Request params: to, amount, note

Response:

Success: {rc: Number, id: String }

Notification: notify to the requested user

Get Money Requests

Name: get_requests

Description:

Request params: date_from, date_to, status, start, size, type

Response:

Success: {rc: Number, money_requests: [req1, req2, ...], total_money_reqs: Number}

Mark Money Request as Accepted

Name: mark_accepted_money_requests

Description:

Request params: receiver_id, request_id, note_processing

Response:

Success {rc: Number}

Mark Money Request as Rejected

Name: mark_rejected_money_requests

Description:

Request params: receiver_id, request_id, note_processing

Response

Success {rc: Number}

Mark Money Request as Cancelled

Name: mark_cancelled_money_requests

Description:

Request params: sender_id, request_id, note_processing

Response

Success {rc: Number}

Mark Money Request as Read

Name: mark_read_money_requests

Description:

Request params: receiver_id, request_ids: Array<{request_id, sender_bare_uid}>

Response

Success {rc: Number}

Blockchain APIs (in progress)

Push transaction to the blockchain

Name: push_transaction

Description: push a transaction raw format (hexa encoding) to the blockchain

Request params: transaction hex

Response: {}

Send token

Name: send_token

Description: send token (coin) to a wallet identified by public address

Request params: to_public_address, amount, message

Response: {}