Fermat

# --draft--

# Fermat

## A Peer-to-Peer Financial Application Framework

**Contributors** : Luis Fernando Molina

**Advisors** :

**Reviewers** :

Noviembre 2014

*www.fermat.org*

## Abstract

A peer-to-peer financial application framework could allow standalone crypto wallets to evolve into any kind of peer-to-peer financial applications.

Developing peer-to-peer financial applications is challenging. Bitcoin provide part of the solution as a p2p system of electronic cash [1], but the main benefits are lost if a trusted third party is still required to transport meta-data, synchronize

devices, hold wallets files or keys, manage identities, interface crypto networks or the legacy financial system.

We propose a peer-to-peer network for transporting meta-data and inter-connect network clients between each other. A synchronization scheme running on top of it transform a standalone app into a distributed application across several devices still owned by the same user.

We propose a framework to replace the standalone wallet application. This framework handles the full stack on top of crypto networks up to the user interface. In this way we enable the development of peer-to-peer financial applications that are both crypto-currency and digital-asset-enabled, and that does not require a trusted third party of any sort.

# Introduction

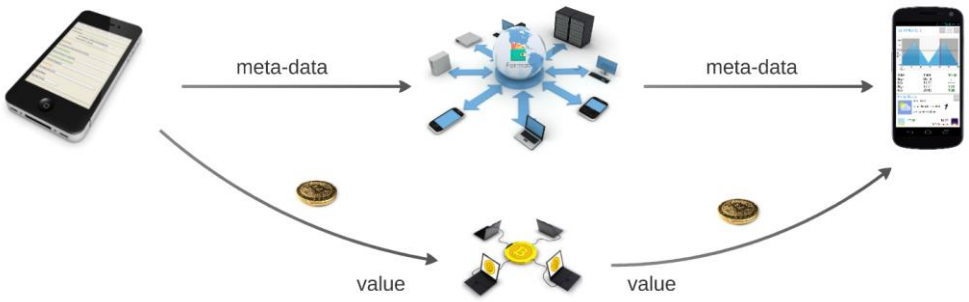
Standalone bitcoin wallets were the first generation of trust-less financial applications since they didn't require to trust any third party, inheriting this property from the bitcoin network itself. As the ecosystem evolved, trusted third parties were introduced again and they took over the wallet space because of technical capabilities that are easier to build in a centralized way: communication between wallets, synchronization between devices, interfacing the legacy financial system, securing funds, etc., and they consistently took the biggest share of funding, leaving standalone wallets far behind and at the same time trashing the benefit of bitcoin of not relying on trust, one of its key features. Applications trying to use the blockchain to transport meta-data were considered spammers and standalone wallets were effectively left behind.

What is needed on top of all existing protocols is a layer that faces the end user and that finishes the job bitcoin started respecting its core principles of openness, decentralization and privacy. Using crypto networks for transporting value or as a registry for digital assets and the Fermat Network for transporting the required meta-data at a network client level, allows financial apps to run any user-level interconnected-functionality without ever going through a trusted third party.

By choosing a plug-in architecture for the Framework we make it possible for any developer to add their own reusable components. We define micro-use-licensing-scheme as the mechanism for plug-in developers to monetize their work. The Framework itself enforces these micro-use-licenses and guarantees developers a revenue stream.

OS dependent GUI components are built on top of the multi-layered plug-in structure to face the end user as wallets or financial applications in general. Apps and wallets with similar functionality are wrapped into what we call*platforms*, each one introducing new plug-ins, to the ever increasing functionality of the whole system.

A built-in *wallet-factory* allows developers to reuse the highest level components and create niche-wallets or niche-financial-apps by combining existing functionality and adding their own code to the combo. A built-in *wallet-editor*allows non-developers to reuse any of these niche-wallets to build new branded-wallets just by changing their look and feel. A built-in *p2p-wallet-store* allows end users to choose which wallets or financial apps to install from the ever growing catalogue.




# The Fermat Network

## Peer-to-Peer Network Architecture

The Fermat Network is structured as a peer-to-peer network architecture on top of the Internet. Fermat nodes are peers to each other, meaning that they are all equal and there are no "special" nodes. All the nodes share the burden of

providing all services. The network nodes interconnect to each other only when they need to do so according to the Fermat P2P Protocol. There is no server, no centralized service, and no hierarchy within the network.

The term "Fermat Network" refers to the collection of nodes running the Fermat P2P Protocol. There are two other protocols such as the Fermat Consensus Protocol which allows the network to agree which transactions are going to be recorded on the blockchain, and the Fermat Client Protocol which is used for coomunicating Fermat Clients between each other and to Fermat Nodes. We use the term "Extended Fermat Network" to refer to the overall network that includes both Fermat Nodes and Clients.

# Fermat Nodes Roles

Fermat nodes performs several tasks at the same time. For each one of them, the protocol has it own set of rules:

## Mantains the Distributed Nodes Catalogue

Each node mantains a full catalogue of all nodes registered in the network. This role is ruled by the Fermat P2P Protocol.

## Mantains an Identities Catalogue

Each node mantains a part of a distributed catalogue of the End User identities. This catalogue is designed to facilitate End Users to find each other. This role is ruled by the Fermat Client Protocol.

## Act as Identities' Home

Each node is home to a set of End Users identities. These identities can receive calls only through their home node. This role is also ruled by the Fermat Client Protocol.

## Acts as a Call Bridge

Each node interconects clients between each other in order to let them freely transfer information between them. This role is ruled by the Fermat Client Protocol.

## Mantains the Fermat Blockchain

Each node mantains the Fermat Blockchain: a public record of all coinbase transactions where the protocol issues new fermats. This role is ruled by the Fermat Consensus Protocol.

### Fermat Clients

Fermat clients run the Fermat Framework, which in turn run the Fermat Components (libraries, add-ons, plug-ins, GUIs, etc.). Clients adhere to the Fermat Client Protocol.

## The Fermat Framework

The solution we propose begins with a Framework that must be portable into different OS. On a multi-layered format, the bottom most layer is interfacing the OS and must be built with replaceable components implementing the same set of public interfaces in order to build on top a single set of OS-independent components. At the same time, the upper most layers are again OS-dependent, providing a native GUI on each device.

We identify 3 different kind of components that we arbitrarily call **Add-ons**, **Plug-ins**, and **GUI** components. We define Add-ons as low level components that do not need to identify themselves to consume services from other components. They have broad access to the file system and databases. Plug-ins have their own identity and must identify themselves to other components to use their services which in return restrict the scope of their services based on the caller's identity (for example the filesystem add-on would only give access to the Plug-ins own folder structure, the database system add-on would only give access to the plugins own databases, and so on). In this way we handle the problem of plug-ins accessing the infromation of other plug-ins.

The core framework is in charge of initializing Add-ons and Plug-ins and managing Plug-ins identities. An internal API library defines the public interfaces that each component exposes to the rest of the components within the same device in order to allow them to use their services locally. This provides a strong encapsulation of each components business logic allowing them to freely define their internal structure and data model.

# Fermat Application Tokens

Fermat generates its tokens, the fermats, with a predetermined algorithm that cannot be changed, and those tokens are necessary for Fermat to function. Fermat miners are rewarded with fermats for their contributions in running the Fermat network.

These application token are native to the Fermat system and are necessary for access to the application. Contribution of value from miners are rewarded in the application’s tokens as well as developers which are rewarded with tokens for the Plug-ins they build for the system. Fermat's blockchain only records the issuing of new fermats and outsources the transaction processing from the bitcoin network. In this sense Fermat is a type II Dapp.[2]


## Token Records

Fermat's tokens data and records are cryptographically stored in a public, decentralized blockchain in order to avoid any central points of failure. This blockchain is stored at Fermat nodes.

## Token Generation & Distribution

Fermat implement three different mechanisms for token generation and distribution:

### Fund-raising Mechanism

With the fund-raising mechanism, tokens are distributed to those who fund the initial development of the Fermat system. The funds collected are used to fund the development of the core of the Fermat system (core libriaries, api libraries, add-ons and the fermat.org web site). The tokens generated during the fund-raising are recorded as the genesis transaction of the Fermat's blockchain.

### Development Mechanism

With the development mechanism, tokens are generated using a predefined mechanism and are only available for the development of Fermat components (plug-ins, GUI components, skins, language packages, and analisys of future development). These fermats become available through a pre-determined schedule and are distributed via a community-driven bounty system where decisions are made based on the proof-of-stake mechanism.

### Mining Mechanism

The Fermat Protocol generates tokens according to a standard crytptographic algorithm acting as a proof of the value nodes are contributing to the application (Fermat uses a kind of Proof of Work Algorithm designed for the particular services Fermat nodes are providing).

With the mining mechanism, tokens are distributed to those who contribute most work to the operation of the Fermat Network. In this case, fermats are distributed through a predetermined algorithm to the miners that connect clients between each other and allows them to talk through them.

## Token Issuing

### Fermat.org (Non-profit organization)

Fund-raising Fermat tokens are issued by a non-profit organization called Fermat.org that will never receive financial benefits from the Fermat system. This organization have the following responsibilities:

- Issuance of initial tokens
- Holding of developer tokens
- Management of bounty payments
- Determining the Fermat System direction

- Collecting and distributing statistical information from the Fermat System.

Fermat.org makes decisions in a decentralized manner, using a “proof of stake” voting mechanism for any decision.

### Fermat Protocol

The protocol itself issues tokens for miners and for development. In the last case, thay are deposited on accounts of the Fermat.org Non-profit until awarded as bounty payments for developers.

## Token Usage

End users automatically aquires fermats by receiving bitcoins into a their Fermat wallet. They can go back to bitcoin by transfering fermats into a Bitcoin wallet.

Fermat tokes are necesary for users to pay for three things:

### Communication

End Users pay Fermat Nodes with fermats to be able to receive calls from other devices.

### Use of Fermat Components

End Users pay Fermat Component's developers with fermats to be able to use their plug-ins, GUI components, skins, language packages, etc. Developers defines a Micro-Use-License for each components. Products like Wallets or Financial APPs use these components, so the cost for using these products is the sum of the cost of the Micro-Use-Licenses defined by each developer involved. This is the way how developers are paid for mantaining their components.

### Technical Support

End Users pay with fermats to receive personalized technical support from Fermat Component's developers.

# Fermat Blockchain

Fermat's blockchain inherits many of the characteristics of the bitcoin blockchain and it is highly coupled with it. The data structure is an ordered, back-linked list of blocks of transactions. In our case all the transactions are coinbase transactions, meaning that they are transactions where new fermats are issued by the protocol. Blocks are linked "back," each referring to the previous block in the chain.

Each block within the blockchain:

- Is identified by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block.
- References a previous block, known as the parent block, through the "previous block hash" field in the block header.

# Mining

Mining is the process by which new fermats are added to the token supply. Mining also serves to the main purpose of the Fermat Network: enable devices to communicate between each other without going through trusted third parties. Miners provide bandwidh to the Fermat network in exchange for the opportunity to be rewarded fermats.

Miners inter-connect devices and acts as a bridge relaying everything from one device to the other. A new block, containing transactions that occurred since the last block, is "mined" every approximatelly 10 minutes, thereby adding those transactions to the blockchain. Transactions that become part of a block and added to the blockchain are considered "confirmed," which allows the new owners of fermats to spend the fermats they received in those transactions.

A transaction at the Fermat Blockchain is considered "irreversible" as soon as it is added to a block. This is true because it is based on information read from the bitcoin blockchain that it is already on an irreversible state. The Fermat Blockchain is syncronized with the bitcoin blockchain but 6 blocks behind the bitcoin blockchain's head.

## Rewards

Miners receive two types of rewards for mining: new tokens created with each new block, and subscription fees from all the network clients that use that node as a home.

### New Minted fermats

To earn this reward, the miners compete to sell incoming bandwith to network clients, i.e. being their home node. Network Clients are free to choose which node to use as their home and at some point they pay in fermats to these nodes for their services. Fermat "proof of work" consist on nodes proving that have received payments for being a home node.

The amount of newly created fermats that can be added to a block decreces approximately every four years. It starts at 50 fermats per block and halves by 2 every 4 years. Based on this formula, fermat mining rewards decrease exponentially until all fermats (21,000,000 million) have been issued. After that, no new fermats will be issued.

### Home Node Fees

Network clients will try to establish their home base at a nearby Node. This will help end users to find the end user behind the network client by knowing approximatelly where he or she lives (city & country). But the network client will scan all nearby nodes and finally decide where to stablish its home base on the plans and tariffs each node is charging for their services.

# Decentralized Consensus

The Fermat blockchain is not created by a central authority, but is assembled independently by every node in the network. The Fermat Protocol provides a set of rules that defines which coinbase transactions are going to be added to the blockchain. As Fermat outsources the transaction processing features of the bitcoin network, it is easier for Fermat to arrive to a consensus.

## Proof of Work

Fermat Proof of Work algorithm is designed in a way to prevent dishonest nodes to lie about the value they are adding to the network.

According to the Fermat Protocol these are the rules to be followed:

- Each node scans the transactions at the bitcoin blockchain block height: head - 6.
- They search for fermat transactions and aggregate all the ones that are payments to nodes in their node catalog.
- They order the node list considering the ones with the highest amount colleted in fees first.
- If they are between the 25% of the nodes that:

a. Received the biggest amount payed by adding all payments.

b. Have been payed with the biggest number of different transactions.

c. The sum of the bitcoin mining fees is the greatest.

Then they are allowed to propose themselves as canditates to receive new fermats by participating in the Race to the Blockchain.

In other words, if the node making all these calculations is at the same time in three different sets of nodes, selected by different criteria, then they should consider themselves candidates for the race and should continue with the next step.

## Race to the Blockchain

Inmidiatelly when a new block is mined at the bitcoin network, the following actions are taken by each qualifying node in order to see if they can earn the new fermats.

They create a coinbase transaction racing between each other to be incorporated first by a bitcoin miner into the bitcoin blockchain at the next block mined. The first 10 % of valid transactions to be incorporated at the bitcoin blockchain will be the ones recorded by every Fermat Node on the Fermat blockchain by adding

them on a new block. The recording will happen when that block has 6 more blocks on top of it.

As every node is reading confirmed bitcoin transactions and they all share a syncronized copy of the node catalogue, the Proof of Work algorithm should give exactly the same result to every node in the network. This means every node knows how many nodes should be part of the race, and how many fermats they should add on their own coinbase transaction in order for the 10% of all these nodes not to exceed the amount of fermats per block.

The sum of the amounts of all these transactions must not exceed the amount of fermats per block allowed by the Fermat Protocol.

## Rationale

By using the fees payed by network clients as "proof of work" we discorage dishonest nodes to lie to the rest of the network about the value added. Network Clients will often pay after the service is delivered. Of course node operators can create fake fees to quilify for the race, but they will need to pay bitcoin mining fees for this and they are not guaranteed that they will win the race to the blockchain. In fact to have better chances to win the race they will have to invest in higher bitcoin miner's fees in order to be included first, again without any guarantee of being among the first 10%.

# Independent Verification of Transactions

In Fermat, coinbase transactions are recorded on the Fermat Blockchain. Previously, nodes recorded the candidate coinbase transactions on the bitcoin network. Those transactions includes the transaction hash on the OP_RETURN field that later is going to be crytical to recognize the satoshis present on the other outputs as fermats by Fermat wallets.

## Fermat Genesis Transactions

We call a *Fermat Genesis Transaction* to each coinbase transaction recorded on the bitcoin blockchain that has also been included on the Fermat blockchain. The structure of that transaction is the following:

| Input # | Contains | Output # | Contains |
|---|---|---|---|
| 1 | bitcoins | 1 | fermats |
| 2 | bitcoins | 1 | fermats |
| 3 | bitcoins | 1 | fermats |
| ... | ..... | ... | ..... |
| n | bitcoins | m - 1 | bitcoins |
| | | m | OP_RETURN |

Note that the *Genesis Transaction* can have *n* number of UTXO as INPUTS, all of them bitcoins (or satoshis to be precise). It can also have *m* number of OUTPUTS where all of them will represent fermats except *m - 1* which is reserved for bitcoin change and the *m* which is used to place the Fermat Coinbase Transaction hash on the OP_RETURN field.

As usual any difference between the sum of all OUTPUTS and the sum of all INPUTS are the bitcoin miner's fees.

So the amount of satoshis on each OUTPUT from *1* to *m - 2* are turned into *fers* that is how we call a ten thousand part of a fermat.

1 fermat = 10,000 fers

We know that by doing this, 1 fermat has a minimun market value of 0.0001 bitcoins.

# Incentive

## For developers

Plug-in developers declare a *Micro-Use-License* for each plug-in they add to the Framework. Wallet or Financial Apps developers declare a *Micro-Use-*

*License* for their components. end users install the Apps (wallets) of their choice. The license to be paid is the summary of the Apps *Micro-Use-License* plus all the *Micro-Use-Licenses* of the plug-ins used by that App.

The Framework is responsible to charge the end user and distribute the payments to all developers involved.

### For network nodes

Network clients establish a *Home Node* where they check themselves and their actors in so as to be found by other network clients. They must pay a subscription fee to their *Home Node* for its services. Finding and calling other clients through other nodes is free for the caller. The nodes income is covered by those network clients for whom they act as their *Home Node*.

## Crypto Networks

A set of Plug-ins is needed for each crypto network to be supported. One for interfacing the network, pushing outgoing transactions and monitoring incoming transactions. Another couple being the digital vaults where the crypto currency value and digital assets are stored.

Wallets are higher level abstractions and have their own set of Plug-ins to keep each kind of accounting. This means that we split the accounting from the handling of the value by having components on different layers to handle each activity.

## Identities

We handle identities at different levels for multiple reasons. In all cases, identities are represented by private and public keys.

## End User Identities

The need to handle multiple logins on the same device brings with it, the first kind of identity which we call *device-user*. This identity lives only at a certain device and not even a public key is exposed to the network.

Besides, the end users can have multiple types of identities (we call this *Actors*), and within each type as many instances as they want. Each type of identity corresponds to a role in real life or an actor in a Use Case. Usually each Platform introduces a set of actors and all the Platforms functionality orbits around all the use cases derived on the interactions between those actors.

The Framework handles a hierarchy of identities. One of them is what we call the *root identity*. At root level end users can set a standard set of information that can be overwritten at any level down the hierarchy, narrowing or expanding that information as needed. All these identities are exposed to the Fermat Network in a way that from the outside, no one could tell they are related between each other or to a certain end user.

## Component Identities

Many components have their identities for a variety of reasons:

a. Plug-ins to identify themselves to Add-ons in order to get access to identity-specific resources, such as Databases or their own share of the File System.

b. *Network Services* to encrypt the communication between each other.

c. Network Clients to encrypt the communication with nodes.

d. Nodes to recognize each other even when their IP location or other profile information changes.

# Platforms

We define as *Platform* a set of interrelated functionality. *Platforms* may consume services from other *platforms* and their dependencies form a hierarchical stack.

Each *Platform* may introduce new workflows to the system , Add-ons, Plug-ins, GUI components (Apps, wallets) and Actors. This enables the system to target different use cases with different actors involved.

# Workflows

We define workflows as high level processes that requires several components to achieve a certain goal. Manny workflows start at a GUI component triggered by the end user and spans through several Plug-ins on the same device, and in some cases jumping into other devices. Other workflows may start at some Plug-ins, triggered by events happening within the same device.

From a workflow point of view, each Plug-in runs a task and is fully responsible for doing its job. Workflows are a chain of tasks that may split into several paths and may span through more than one device.

In some cases workflows interconnect with each other, forming a *workflow chain* that usually spans more than one*Platform*.

# Transactions

## Transactional Workflows

As the Framework runs on potentially unstable devices such as mobile phones, each Plug-in must be prepared to overcome the difficulties caused by a device shutting down at any moment and it must be able to complete its intended job later and never leave information on an inconsistent state. This is quite challenging but not impossible.

The solution is to make each Plug-in responsible for the workflow while they are handling part of a transaction on a transactional workflow. This responsibility is transferred to each step of the chain using what we call a *Responsibility Transfer Protocol*. This means that the component that is responsible at the moment of a black out is the one that must resume and do its best to get rid of that responsibility moving it further down the chain within the transactional workflow.

## Value Transactions

We handle monetary and digital assets transactions dividing the accounting from the value. Usually transactions start on specialized Plug-ins which are in charge of coordinating the whole transaction. These Plug-ins usually interact with wallets-Plug-ins debiting or crediting the accounts involved. The accounting of the currency or digital asset involved are kept by these wallet-Plug-ins. Later the transactional workflow splits between moving the value (usually crypto currency) and moving the meta-data associated to the transaction.

Through two different paths, the value and the meta-data arrives to the recipients device and they are combined together by the remote counter-party transaction component which in return interacts with the remote wallet-Plug-in to record the accounting as appropriate.

# Synchronization

We define a Private Device Network as a network of devices owned by the same end user. Using the Fermat Network, the Framework synchronizes the information on all nodes of the Private Network. By doing so the information and system identities belonging to the end user are available at any device.

Crypto funds are kept into a Multi-Sig vaults and there they are shared, making *Petty-Cash-Vault* accessible from all nodes even when they are offline from this Private Network. An automated process monitors the Petty-Cash-Vault and tops it up when needed. Several nodes must sign the top transaction in order to proceed. This way if a device is lost or stolen, only the Petty-Cash fund is at risk. End users can eject stolen devices from its Private Network and if they act quickly they might be on time to re-create the Petty-Cash fund under the new configuration and be able to save those funds.

# User Interface

The Framework handles a stack of layers. Starting from the bottom we have the *OS API level*, then the *Blockchain Level*, the *Communication Level*, *Platform Level* and the *User Interface Level*. With the goal in mind for allowing even non-developers to deploy their own peer-to-peer financial applications, we define several concepts:

**Wallet**: Any kind of financial application that handles either crypto or digital assets for any purpose.

**Reference Wallet**: A primitive wallet that is used by a single actor for a handful of use cases.

**Niche Wallet**: A combination of several *Reference Wallets* into a single product with its own look and feel and possibly extra functionality.

**Branded Wallet**: A *niche wallet* turned into a new product owned by a different end user. Achieved by a process similar to building a Wordpress site but locally, on the end users device. Usually it involves re-using the business logic of the *niche wallet* it derives from and adding a new look and feel (different skin and navigation structure).

**External Wallet**: A third party APP running on the same device that uses Fermat as a backend for different reasons. For example to benefit from its infrastructure to interface crypto networks, transporting data through its p2p network, or storing data on the end users *Private Device Network*.

Several tools were designed with the purpose of enabling the development of new wallets, an their distribution.

**Wallet Factory**: Is a built-in functionality that enables the development of reference and niche wallets.

**Wallet Editor**: Enables the creation by non-developers of *Branded Wallets* based on any one of the *Niche Wallets*available.

**Wallet Store**: Is a distributed application which manages a shared wallet catalog and enables the end user to download from peers the different wallets available for the Framework.

# Privacy

The proposed system complements the privacy properties of crypto networks, extending them to the full stack needed to run different kind of financial applications. By using its own P2P network with point to point encryption for transporting meta-data both value and information are under a similar privacy standard.

Identities are public keys related to private keys kept by the end user and never shared to anyone in any way.

The collection of system information for visualization and statistics uses hashes of public keys to protect end users privacy and at the same time preserve the relationships between them.

# Conclusion

We have proposed a system for developing and running peer-to-peer financial applications. The Fermat Framework shows the way of how to keep the end user away from trusted third parties at a higher level. We propose a solution to several problems at the same time. The highlights of our work are:

- How to exchange meta-data in a peer-to-peer way
- How to prevent the loss of private keys (funds and identities)
- How to maximize reusability by building with Plug-ins
- How to enable even non-developers to create and use their own wallets and financial applications.

With this system we allow for a new ecosystem of peer-to-peer financial applications that are both crypto and digital asset enabled.

# References


- [1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", https://bitcoin.org/bitcoin.pdf, 2008
- [2] David Johnston & others, "The General Theory of Decentralized Applications, Dapps",https://github.com/DavidJohnstonCEO/DecentralizedApplications/blob/master/README.md


# Further Reading

- DRAFT: Fermat Book - https://github.com/bitDubai/fermat/tree/master/fermat-book