

# SpiderByte White Paper: A Better Peer-to-Peer Electronic Cash System



Simone Lussardi

[spider@spiderbyte.co](mailto:spider@spiderbyte.co)

[tysimon@spiderbyte.co](mailto:tysimon@spiderbyte.co)

[help@litecoinplus.co](mailto:help@litecoinplus.co)

[media@litecoinplus.co](mailto:media@litecoinplus.co)

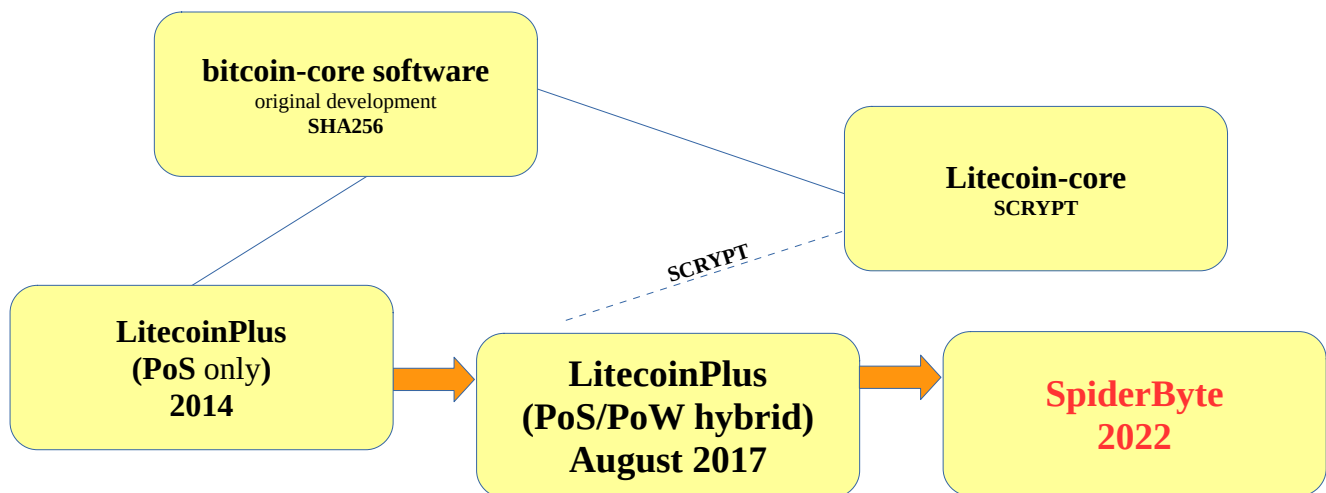
<https://spiderbyte.co/>

(git: <https://github.com/Crypto-Currency/SpiderByte>)

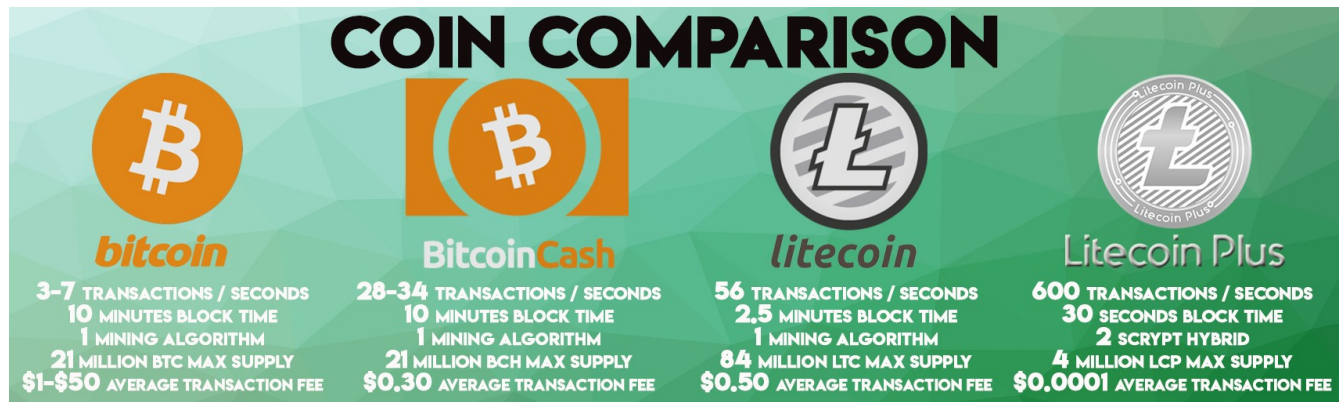
[rev 5.00 - 2023-02-01 by tysimon]

[re-brand from Litecoin Plus / LCP]

**Abstract.** A purely peer-to-peer version of electronic cash to allow online payments to be sent directly from one party to another without going through a financial institution, while holding a more or less stable value. Basic project is derived from Bitcoin (BTC) software and Litecoin (LTC), has the same base structure and functionality, but it is an improvement under several aspects. First and foremost, SpiderByte team believes that PoW (Proof-of-Work, and related massive carbon print and power consumption) is destined to be left behind. It is no doubt necessary to maintain a fast moving network growing. The following charts clarify the evolution of SpiderByte (former LitecoinPlus):



To put in graphical terms the fundamental differences between each of the above systems, here is a simple chart:



There is no need to re-invent the wheel and write down the specifics of Bitcoin or Litecoin, since about everyone knows about those. If you are interested in the mathematics behind Bitcoin, block generation etc. (which applies fully to SpiderByte network as well), please refer to the original white paper of Satoshi Nakamoto: <https://bitcoin.org/bitcoin.pdf>. This white paper instead is about how we achieved the target mentioned above and how we plan to sustain it through many years to come.

## 1. Introduction

SpiderByte has not started via an ICO (Initial Coin Offering). It was originally written as Proof of Stake coin exercise. *“Proof of stake (PoS) is a type of algorithm by which a cryptocurrency blockchain network aims to achieve distributed consensus. In PoS-based cryptocurrencies, the creator of the next block is chosen via various combinations of random selection and wealth or age (i.e., the stake). In contrast, the algorithm of proof-of-work-based cryptocurrencies such as Bitcoin uses mining; that is, the solving of computationally intensive puzzles to validate transactions and create new blocks”* (ref <https://en.wikipedia.org/wiki/Proof-of-stake>). In contrast to the better known PoW, the non-computational, consensus nature of the algorithm make sure that power consumption is not an issue. PoS, when applied properly, can be run on any type of hardware, even very old computers just running the SpiderByte wallet software can generate enough blocks to sustain the network.

The fundamental issue of the first release of SpiderByte in 2014 was that it was a pure PoS coin, and as such an experiment. In order to sustain the network, a stable block generation is required (see later), which led to the original SpiderByte network not able to sustain itself.

In the next page, we put some generic details on the specifications of SpiderByte.

# GENERAL SPECIFICATIONS

Name: SpiderByte

Acronym: SPB

Encryption: Scrypt

Block generation: Proof of Stake/Proof of Work hybrid

Transaction time: 30 seconds

Max supply: 4,000,000 coins

RPC Port: 44350

P2P Port: 44351

## Some resources for preliminary overview:

Main website: <https://spiderbyte.co>

Web wallet: <https://wallet.spiderbyte.co>

Stable git: <https://github.com/Crypto-Currency/SpiderByte>

Development/testing git: <https://github.com/typhoonsimon/LitecoinPlus>

Bitcoin Talk: <https://bitcointalk.org/index.php?topic=2160325.0>

Cryptocurrency Talk: <https://cryptocurrencytalk.com/topic/87530-ann-litecoinplus-wallet-update-pow-added/>

Twitter: [http://twitter.com/Media\\_SPB](http://twitter.com/Media_SPB)

Discord community: <https://discord.gg/UqXEmd5>

## Proof of Work specifications

Start reward at 2 and reduce by .25 every 200,000 blocks, for a duration of roughly 9,6 years (1 block every 30 seconds is the global target), until reward reaches 0.5 SPB, then it will stay at 0.5 SPB/block by default.

**NOTE:** From the start of PALADIN, that is planned to commence in early March 2020, the Proof of Work reward will be gradually reduced, and possibly fade out if the results of the experiment are good.

## Proof of Stake specifications

Interest rate: 15% annually

Minimum coin age: 6 hours

Maximum age: 30 days

**NOTE:** From the start of PALADIN, the PoS reward may be dynamically increased from time, to time to stimulate more nodes to participate.

# Compiling

Tested versions: 3.3.2.9~5.0.0.1

(\* only LTS versions, fully updated systems)

(\*\* all versions are fresh installs, other combination of libraries may not work)

OS	Install sequence (depen.)	Server	QT
Ubuntu 12.04 32/64	<pre>sudo apt-add-repository ppa:bitcoin/bitcoin sudo apt-get update sudo apt-get install libdb4.8-dev libdb4.8++-dev &gt; broken (precise support removed from bitcoin/bitcoin ppa) &gt; can use another version of Berkeley, but this will broken compatibility, so is strongly <u>not</u> recommended</pre>	Obsolete	Obsolete
Ubuntu 14.04 32	<pre>sudo apt-get install build-essential sudo apt-add-repository ppa:bitcoin/bitcoin sudo apt-get update sudo apt-get install libdb4.8-dev sudo apt-get install libdb4.8++-dev sudo apt-get install libboost-all-dev sudo apt-get install git sudo apt-get install libssl-dev sudo apt-get install libqt4-dev git clone https://github.com/Crypto-Currency/SpiderByte.git cd SpiderByte  :::QT qmake USE_UPNP=- make  :::d make clean cd src/ make -f makefile.unix USE_UPNP=- strip ./spiderbyte</pre>	Fully tested	Fully tested
Ubuntu 14.04 64	[exactly same as 14.04 32]	Fully tested	Fully tested
Ubuntu 16.04 32	[exactly same as 14.04 32]	Fully tested	Fully tested
Ubuntu 16.04 64	[exactly same as 14.04 32]	Deep tested	Deep tested
Ubuntu 18.04 64	[exactly same as 14.04 32] must install Gnome Extension "KStatusNotifierItem/AppIndicator Support by 3vin0" to enable tray icon support	Deep tested	Deep tested
Debian (all versions)	[same as Ubuntu, but cannot use libdb4.8, unless one compiles it by him/herself]	Fully tested	Fully tested
Fedora 29~31	[procedure coming up, everything working so far, including migrating data from Ubuntu to Fedora]	Fully tested	Fully tested
Windows XP	[just run the setup software]	Fully tested	Fully tested
Windows 10	[just run the setup software]	Fully tested	Fully tested

"Fully tested" sequence (qt means graphic version, d means server version):

1. Start the software (-qt or d) completely fresh, creates all the files, start syncing properly. After a while shutdown. Start again, can start and continue where it left off.
2. Replace the folder with existing data folder (for respective bits), start the software, make sure it can run properly.
3. Staking on -qt version
4. Transferring coins from/to other wallets
5. PoW mining on -qt and d version
6. Alternating between -qt and d versions
7. Partially import a bootstrap file, then continue via the network
8. Easily maintaining in sync once fully synced (no stuttering)
9. Long time stability (no freezes, no abrupt terminations)

"Deep tested" sequence:

1. All of the above sequence, plus the following items:
  1. Import completely a bootstrap file successfully.
  2. PoS mining on -qt version
  3. PoW mining on -qt and d version

## 2. Bitcoin: areas of improvement

Bitcoin has forever changed the world view on how “value” can be exchanged, opening an entire new area into the crypto-currency world. As the first of anything, great innovation was brought to us by very fact of the market capitalization it achieved today.

Having said that, Bitcoin has a few points that prevent it from becoming widespread as a quick payment method:

- **Confirmations take a long time:** on average, payment confirmation may take up to 1 hour to get through. This has made Bitcoin to be essentially used as a trading commodity only instead of allowing online payments.
- **Transfers are expensive:** given the popularity and price increase, transaction fees have become particularly expensive for Bitcoin.
- **Energy intensive:** getting the Bitcoin network going and the necessary block generation is an expensive business.

SpiderByte has improved on all the above points, with careful planning of the revised release in August 2017.

## 3. Hybrid concept

As the original failure has shown, a pure PoS network is not able to sustain itself, especially at the beginning. Since there was no ICO and no initial coins (except for those generated in the initial stages of the tests), it was impossible to get the amount necessary to get the net going. For this reason, PoW was reintroduced, with a gradual “phase out” scheme, as below:

- **Original tests:** reward of 2.00 SPB per block, plus transaction fees
- **Block # < 2,000,000:** reward of 1.75 SPB per block, plus transaction fees
- **2,000,000 <= Block # < 2,200,000:** reward of 1.50 SPB per block, plus transaction fees
- **2,200,000 <= Block # < 2,400,000:** reward of 1.25 SPB per block, plus transaction fees
- **2,400,000 <= Block # < 2,400,000:** reward of 1.00 SPB per block, plus transaction fees
- **2,600,000 <= Block # < 2,400,000:** reward of 0.75 SPB per block, plus transaction fees
- **2,800,000 <= Block #:** reward of 0.50 SPB per block, plus transaction fees

This reward, combined with a block generation target of 30s, will be used to calculate the following two parameters:

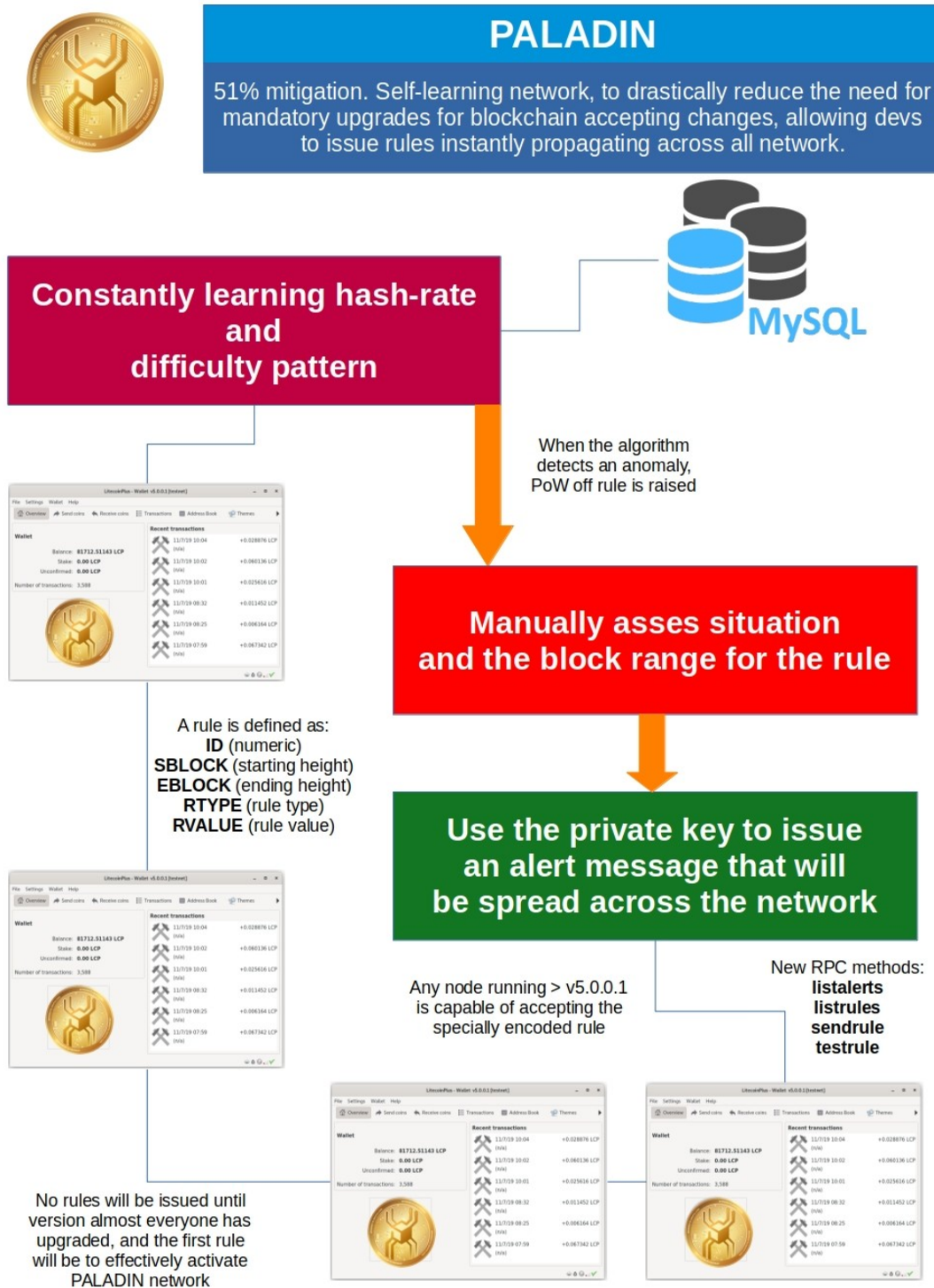
- PoS difficulty
- PoW difficulty

The aim of these two parameters is to balance out new block generation, and achieve the target specified of 30s. Blocks are publicly readable at: <http://explorer.spiderbyte.co/> or <http://explorer2.spiderbyte.co/>

At the top of the block explorer, real time PoW and PoS difficulty, current supply and Bitcoin price are available.

# 4. PALADIN Implementation

PALADIN is a self-learning advanced system, that has the following scheme:





As of version 5.0.0.1 or newer of the wallet, PALADIN system has been introduced. PALADIN is a 51% attack mitigation set of rules and functions that can dynamically change the behavior of the entire network, remotely and with consensus. This approach is quite innovative and it maintains fully the decentralized structure and nature of SPB, in the true spirit of the original white-paper. Actually eliminating the 51% problem in a true decentralized scheme is not fully possible as of today, unless central control is introduced, and after much discussion and deliberation, the development team has come to the conclusion that centralizing the control with a server shouldn't be done.

But we want to assure all of SPB investors that we have a strong plan for mitigation, should a problem ever arise. Our concept is based on simple assumption:

1. PoS mining is intrinsically a lot less likely to have an attack mounted, due to the nature of how blocks are generated and checked
2. SPB is already an hybrid crypto-currency, and we feel that adding another mining method to it would not be a good option

We come to the conclusion that the attack may only come from PoW mining, and for this reason, we have adopted several mechanism since version 5.0.0.1.

From new file **rules.h**:

```
// the PALADIN system class definition
class CRules
{
public:

    // saving stuff for the rules/alert (encrypted only)
    std::vector<unsigned char> vchMsg;
    std::vector<unsigned char> vchSig;

    // variables
    int alertId; // link to alert ID
    int nVersion; // version of rule packet
    int nID; // rule ID
    int nMinVer; // lowest version inclusive
    int nMaxVer; // highest version inclusive
    int fromHeight; // starting from block height
    int toHeight; // ending at block height
    int ruleType; // just an enum of rule types
    int ruleValue; // the value for this rule
    enum ruleTypes
    {
        RULE_POW_ON_OFF = 1, // PoW ON/OFF flag
        RULE_CLOCK_DRIFT = 2, // adjust nMaxClockDrift value dynamically
        RULE_POS_PERCENT = 3, // adjust PoS % reward
        RULE_POW_REWARD = 4, // adjust PoW absolute reward
        RULE_BLOCK_TARGET = 5, // adjust the block target, in s
        RULE_DISABLE_OLD_CLIENTS = 6, // only accept clients >= 5.0.0.1
        RULE_SUSPEND_SENDING = 7, // refuse to send coins, globally
        RULE_POS_ON_OFF = 8, // PoS ON/OFF flag
    };
};
```

1. **RULE\_POW\_ON\_OFF**: with this rule, we can control whether the network, within a specific range of blocks, accepts PoW or not altogether. If the rule is raised, PoW blocks are rejected by the network, and the node is soon disconnected by the clients, as the default behavior for invalid PoW found.
2. **RULE\_CLOCK\_DRIFT**: an attack is often carried by advancing the nTime of blocks rapidly mined, so that other nodes are too busy re-organizing the chain, and creating mini forks. Is not necessarily a dangerous thing, just annoying, cause the users would then need to re-sync the blockchain. By adjusting this value dynamically, we can fine tune it if this is detected by our algorithm.
3. **RULE\_POS\_PERCENT**: possibility to adjust the % of PoS remotely. Not really directly used against the attack, but combined with PoW off under certain conditions, could be used to achieve a drastic increase in PoS mining to sustain the network.
4. **RULE\_POW\_REWARD**: possibility to adjust the absolute reward of PoW mining. Again, not directly linked, but since we have added PoS dynamic percent, why not adding also PoW absolute reward. It would probably never be used, but is there.
5. **RULE\_BLOCK\_TARGET**: dynamic block target value (at the moment is 30s). Could be used also to mitigate an attack situation by diluting slightly the block generation.
6. **RULE\_DISABLE\_OLD\_CLIENTS**: essentially is the “PALADIN ACTIVATED” flag, and does a few things internally in the code, among which, enabling PALADIN.
7. **RULE\_SUSPEND\_SENDING**: when this rule/flag is activated, all legitimate wallets will be unable to send coins out. This function is essentially the equivalent of suspending withdraws for Exchanges or mining pools or web wallets when a serious problem is detected, and precautionary, stop them. Receiving cannot be stopped, but an exchange could check the status of this rule dynamically with the new RPC command “testrule <height> <rule-type>”, in which <height> is the block height to test, and <rule-type> the rule type required to test, in this case 7. Exchanges could then take immediate actions in case they are receiving coins from some nodes, because legitimate nodes shouldn't be sending when the rule is on, meaning very likely that the system is being targeted by malicious transactions.
8. **RULE\_POS\_ON\_OFF**: with this rule, same as above, we can lock the acceptance of PoS blocks. Along with PoW flag above, with this rule we can effectively pause the blockchain, without adverse effects on connectivity. Of course, as above, these commands are expected to be used in emergency.

#### 4.1 - How are rules activated ?

We have profoundly improved the alert management sub-system, already present with modules `alert.cpp/alert.h`. We have also vastly improved the RPC part of the “sendalert” method. Rules are activated by a new RPC method, “sendrule”, that has the following parameters:

- **<privatekey>**: is hex string of alert master private key. Only the main devs have this key.
- **<encoded-rule>**: the encoded string that defines the rule.



Because a new RPC method has been created, it is not necessary to use “sendalert” to encode rules. Also, this is prevented, as rule are specially encoded alert with priority = 999, set in the code.

#### 4.2 - How are rules are spread ?

Once activated, these specially packed alerts have a flag nPermanent set, and they can never be canceled. Even old nodes are able to decode these packets, and they will relay them, but they will not accept them because the protocol version match has been increased.

They will live permanently on the net. As soon as a rule is received, it is stored in memory and the equivalent encoded packet (class: **Calert**) it is stored to the disk, in a sequential format, in a file called rules.dat.

In the unlikely event that all the nodes are down simultaneously, upon restart the first node will spread its saved rules to the network. A node having fresher rules will do the same, and the combination will result in a unique, updated set of rules. Because rules and blockchain goes together, if a block is ever in the blockchain, it means it was accepted by that very node that has the rule, so at least one copy of the freshest rule will always be present in at least one node.

Of course, the Block Explorer, Website and Web-Wallet nodes, even though they are not super-nodes or anything as such, will always have the latest set of rules just because their uptime is guaranteed by the devs.

### 4.3 - If someone corrupt the rules.dat file ?

Because we save the encrypted packet that originated the alert, and decryption is done every time the file is read again, it is extremely unlikely to happen. It has the same chances of discovering the private key, but at that point corrupting the file is no more necessary, as the malicious user would spread messages across the network.

For that purpose, there is a built-in safety, that allow the devs to disable entirely the acceptance of new alerts. It is a specially encoded alert that will effectively lock down the system, telling the users to upgrade:

```
// alert.nID=max is reserved for if the alert key is
// compromised. It must have a pre-defined message,
// must never expire, must apply to all versions,
// and must cancel all previous
// alerts or it will be ignored (so an attacker can't
// send an "everything is OK, don't panic" version that
// cannot be overridden):
int maxInt = std::numeric_limits<int>::max();
if (nID == maxInt)
{
    if (!(
        nExpiration == maxInt &&
        nCancel == (maxInt-1) &&
        nMinVer == 0 &&
        nMaxVer == maxInt &&
        setSubVer.empty() &&
        nPriority == maxInt &&
        strStatusBar == "URGENT: Alert key compromised, upgrade required"
    ))
        return false;
}
```

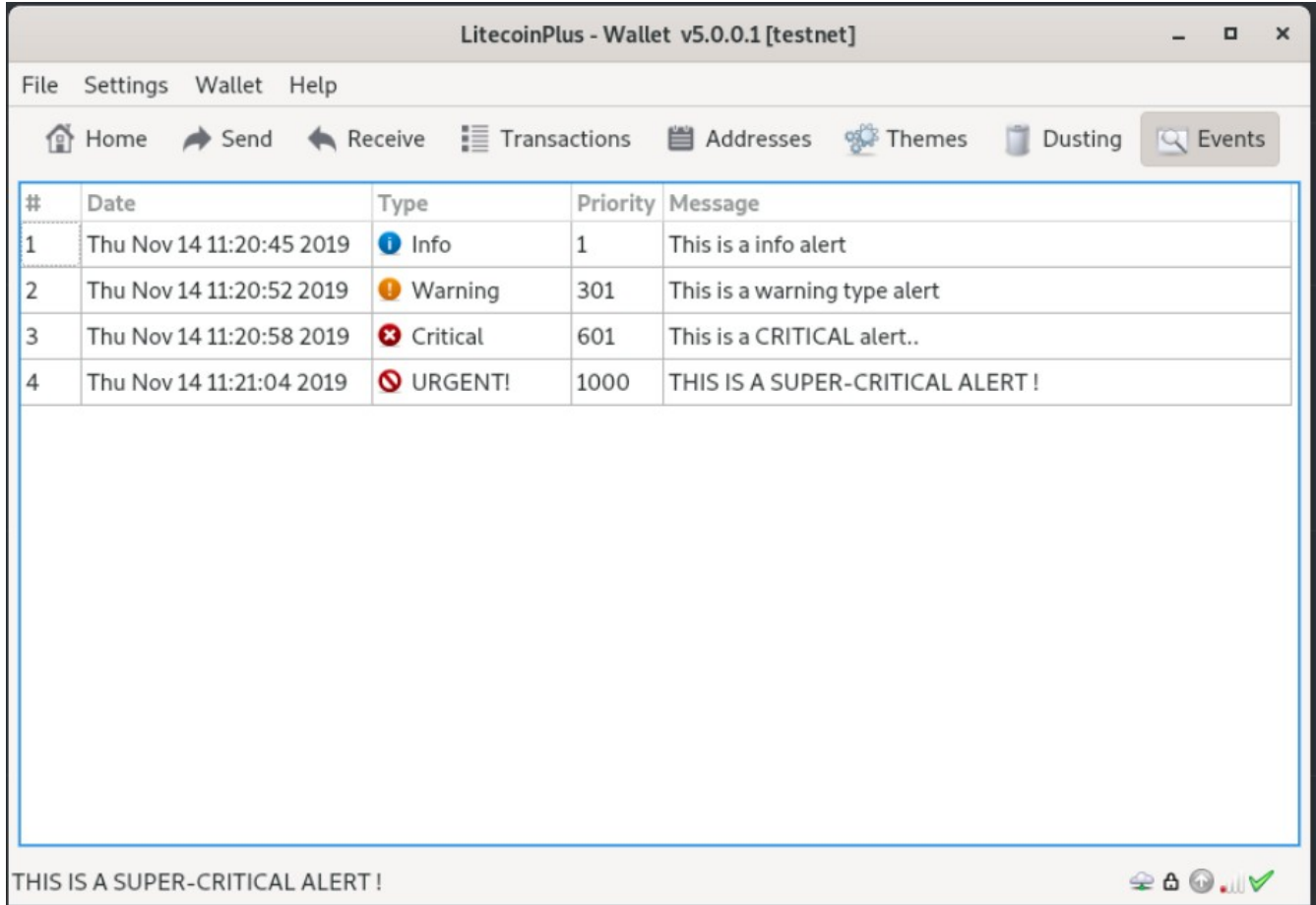
### 4.4 - Upgraded RPC interface

The RPC interface has been enriched by the following commands:

- **sendrule**: transmit a rule to the network.
  - **<privatekey>**: is hex string of alert master private key. Only the main devs have this key.
  - **<encoded-rule>**: the encoded string that defines the rule.
- **listrules**: lists all rules that are accepted and stored in memory
- **testrule**:
  - **<height>**: the block height to test (one can go in the past and future)
  - **<rule-type>**: the rule type # required, according to the ENUM structure above
- **listalerts**: list all active alerts in memory.

## 4.5 - New improved alert system

A new improved alert system has been released alongside the rules, so that wallet users can receive important information at any time, for QT users in a specific newly designed “Event” page. The page looks like below:



The screenshot shows the LitecoinPlus - Wallet v5.0.0.1 [testnet] interface. The main window displays the "Events" page, which contains a table of alerts. The table has five columns: #, Date, Type, Priority, and Message. The alerts are as follows:

#	Date	Type	Priority	Message
1	Thu Nov 14 11:20:45 2019	Info	1	This is a info alert
2	Thu Nov 14 11:20:52 2019	Warning	301	This is a warning type alert
3	Thu Nov 14 11:20:58 2019	Critical	601	This is a CRITICAL alert..
4	Thu Nov 14 11:21:04 2019	URGENT!	1000	THIS IS A SUPER-CRITICAL ALERT !

At the bottom of the window, a status bar displays the text "THIS IS A SUPER-CRITICAL ALERT !" and several system icons.

The categorization is done in the following way (`alert.h` and `alert.cpp`):

```
/** An alert is a combination of a serialized CUnsignedAlert and a signature. */
class CAlert : public CUnsignedAlert
{
...

    static bool isInfo(int priority);
    static bool isWarning(int priority);
    static bool isCritical(int priority);
    static bool isSuperCritical(int priority);
    static bool isRule(int priority);
};

...

bool CAlert::isInfo(int priority)
{
    return (priority <= 300);
}

bool CAlert::isWarning(int priority)
{
    return (priority <= 600);
}

bool CAlert::isCritical(int priority)
{
    return ((priority > 600) && (priority < 999));
}

bool CAlert::isSuperCritical(int priority)
{
    return (priority >= 1000);
}

bool CAlert::isRule(int priority)
{
    return (priority == 999);
}
```

The type itself is pretty much self explaining. The only difference is that only SuperCritical category alerts appear in the status bar, all the other will appear only in the new view or in the “listalerts” RPC method.

SuperCritical category has a special case also: priority 1000. It is possible to send a SuperCritical alert that will not appear in the status bar.

As per  $> 1000$  ( $\geq 1001$ ), not only the status bar will show the message, but also the wallet will spontaneously enter SAFE MODE, not allowing a lot of functions to happen. Obviously this alert/rule would be used only in severe cases that may compromise the safety of our users.

## 5. Mining in details

Because SpiderByte has a hybrid bloc generation, mining takes place in two separate ways:

1. PoS Mining
2. PoW Mining

Regarding **PoS Mining**, it can be done by anyone on any type of hardware. It is recommended to have a stable Internet connection. The way of block generation is as following:

1. The wallet confirmed balance must be  $\geq 0$
2. If the wallet is encrypted, upon start it needs to be unlocked for staking (the icon indicating the staking status will report about this, and the wallet can be safely unlocked by going to menu **Wallet** → **Unlock To Stake...**).
3. Unlocking to stake is safe, the wallet will still need to be unlocked if any send transaction is required.
4. The algorithm will then mature coins for staking.
5. When the wallet is actively staking, the staking icon will turn green.
6. When a stake is generated, the following will happen:
  1. The transaction will appear in Overview and Transaction pages as unconfirmed
  2. The block that staked is divided into two, and one part is put into stake status (not spendable)
  3. When it starts to get confirmations, by hovering the mouse on the transaction, you can see the number of confirmation received
  4. When it received 3 confirmations, the block reward is **confirmed but not spendable** yet.
  5. When it matures 50 or more confirmations, the related reward become spendable and the stake amount is released, so it can mature again.

This is the calculation code for the PoS Mining reward:

```
int64 GetProofOfStakeReward(int64 nCoinAge, unsigned int nBits, unsigned int nTime, int nHeight)
{
    int64 nRewardCoinYear;
    if(nHeight < STAKE_FIX_BLOCK)
    {
        nRewardCoinYear = MAX_MINT_PROOF_OF_STAKE;
    }
    else
    {
        nRewardCoinYear = MAX_MINT_PROOF_OF_STAKE2;
    }
    int64 nSubsidy = nCoinAge * nRewardCoinYear / 365;
    if (fDebug && GetBoolArg("-printcreation"))
        printf("GetProofOfStakeReward(): create=%s nCoinAge=%"PRI64d" nBits=%d\n",
            FormatMoney(nSubsidy).c_str(), nCoinAge, nBits);

    return nSubsidy;
}
```

Regarding **PoW Mining**, any equipment that can handle SCRYPT algorithm mining (cpuminer, cgminer or ASIC miners such as AntMiner L3+, InnoSilicon A4+ etc.) can be used for SpiderByte mining. As difficult increases, CPU or GPU mining is not profitable using anything else than ASIC miners. The Principle is as follows:

1. With given difficulty (block target = 30s), ASIC miners will generate shares
2. When a block is found, the block is submitted to the network via RPC
3. If accepted, the block will appear in the wallet as unconfirmed
4. Once 3 confirmations are received, the block is marked as confirmed, and the reward as immature, going to the immature balance, thus **not spendable**.
5. After **50 or more blocks**, the amount becomes mature and thus **spendable**.

Here below is the code calculating the PoW reward:

```
int64 GetProofOfWorkReward(int nHeight, int64 nFees, uint256 prevHash)
{
    int64 nSubsidy = 0 * COIN;

    if(nHeight == 1)
    {
        nSubsidy = 221400 * COIN;
        return nSubsidy + nFees;
    }

    if(nHeight > POW_RESTART_BLOCK)
    {
        nSubsidy= 2 * COIN;
        if(nHeight > POW_RESTART_BLOCK+200000)
        {
            nSubsidy = 1.75 * COIN;
        }
        if(nHeight > 2000000)
        {
            nSubsidy = 1.5 * COIN;
        }
        if(nHeight > 2200000)
        {
            nSubsidy = 1.25 * COIN;
        }
        if(nHeight > 2400000)
        {
            nSubsidy = 1 * COIN;
        }
        if(nHeight > 2600000)
        {
            nSubsidy = .75 * COIN;
        }
        if(nHeight > 2800000)
        {
            nSubsidy = .5 * COIN;
        }
    }
    return nSubsidy + nFees;
}
```

For all mining, expect some orphans to happen occasionally, especially for PoS mining, since not all the blocks will be able to make it into the network, especially if they are generated very close to each other (within a few seconds).

## 6. Coin control and Dusting

Because of the nature of PoS Mining, and block split, it's just a consequence that the number of blocks that build up the wallet balance will exponentially increase over time. This problem may become an issue for the user for two reasons:

1. When attempting to send a big amount, if the balance is built up by many tiny blocks, the sending function may fail even though the balance is enough.
2. The smallest fraction of coin is 8 decimals (0.00000001, for instance). When the block become so small that the related stake reward is less than that, the wallet will not stake anymore.

To resolve the above issue, the user has two options:

1. Using the Advanced Coin Control features (**not recommended** for normal users), the user can recombine his/her own coins into single larger block, so they will stake again.
2. Using the automatic Dusting function the internal algorithm will just optimize the blocks so that the wallet can continue to fairly stake and sustain the network, while not causing problem to the user. In order to dust:
  1. Select a destination address (**only own addresses can be chosen**)
  2. Press the “**Dust now**” button

In either cases, it may appear that after dusting, the wallet stake “*less than before*”. As per frequency this is true, but the amount at the end of long period is exactly the same, if not more.

Here below is a sample page of the Dusting function, simply press **Dust now** button:

Amount	Date	Label	Received with	Confirmations
0.00009	18-06-03 10:39	(change)		10196
13.66	18-06-03 12:25	[DUSTING]		9879
13.674646	18-06-03 12:25	[DUSTING]		9879
15.61	18-06-05 10:02	[DUSTING]		76
15.625589	18-06-05 10:02	[DUSTING]		76
45.71	18-06-04 12:32	[DUSTING]		4740
45.71	18-06-04 12:47	[DUSTING]		4687
45.71	18-06-04 07:29	[DUSTING]		5809
45.71	18-06-04 12:06	[DUSTING]		4845
45.71	18-06-04 08:02	[DUSTING]		5711
45.71	18-06-04 12:28	[DUSTING]		4782

Found 30 blocks to compact.





## 8. Maximum supply

One of the target being the stable value, the maximum supply of coins is limited to 4,000,000 coins. Once that amount is reached, the software will not generate any more reward. The reward for generation of blocks will come exclusively from transaction fee. By then (we estimated roughly 10 years) the amount of transaction will be enough and the value would have risen to an amount that will sustain the network.

## 9. PoS and staking

In order to get the PoS going, the wallet need to have coins in it. A wallet with zero coins does not stake, thus it doesn't generate any blocks nor rewards. Once a positive balance is in, coins will need to "mature" in order to start generating rewards. It is all part of the internal algorithm to decide what blocks/coins mature and stake. The guaranteed annual return for staking is about 15%.

## 10. Reaching *all* users

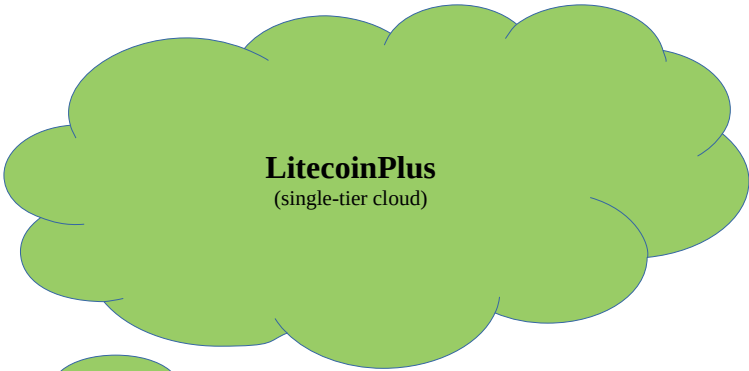
SpiderByte has considered carefully all kind of users, from the IT expert to the non-expert. For this reason, a comprehensive structure of wallets have been devised, as following:

- **Software wallet:** the software wallet is the only one capable of generating blocks, in combination with mining equipment or stand-alone. Several versions are available:
  - Windows version
  - Mac OS version
  - Linux versions
    - QT version (spiderbyte-qt, intended for Linux graphical interface users)
    - *d* version (spiderbyted, intended for mining professionals and exchanges)
- **Web wallet:** easily accessible via web, it is already active and running at <https://wallet.spiderbyte.co>. This makes it accessible to a large audience.
- **Android wallet:** even more easy than the web wallet, it's available on Google Play at the following page: <https://play.google.com/store/apps/details?id=com.litecoinplus.androidwallet>

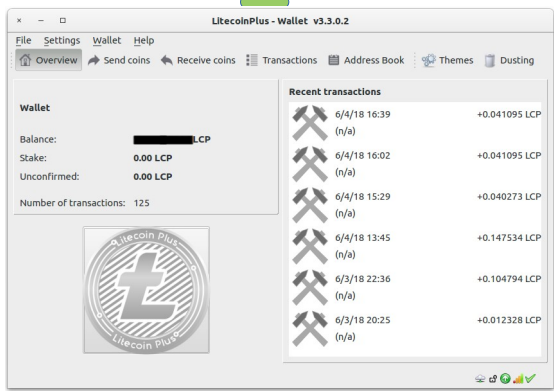
While all software flavors are basically the same, their behavior slightly differs. For example, when installing the daemon "d" version for exchanges or mining portals, the staking functionality needs to be disabled, otherwise funds cannot be transferred (see "PoS and staking"). In the next page, there is a chart displaying how the single-tier network of SpiderByte works with the respective type of wallets.

Staking is possible an all wallets. For the Web and Android Wallets, current percent is set to 10% and paid weekly in the user wallet. Minimum deposit has to be 100 SPB, and 5 days age at least.

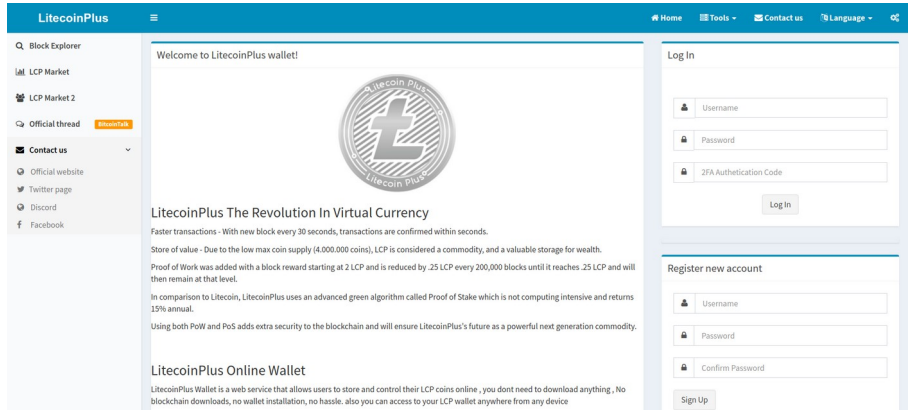
*Relationship between the single-tier cloud and the user experience*



**LitecoinPlus**  
(single-tier cloud)



**Wallet software**



**Web Wallet**



**Android Wallet**



The advantages of the above solutions are obvious, and all aimed at simplifying the user experience. For those more advanced users who prefer to own their coins into their devices, the traditional wallet software is always available.

## 11. Software architecture

With its latest official release version 3.3.2.12, the wallet control has established the following principles and philosophy for its architecture and development:

1. **Database engine Berkeley DB (currently 4.8):** contrary to the majority of other wallets, even popular ones like Bitcoin and Litecoin, we believe the extreme solidity and reliability of Berkeley DB is not worth the change. Most of them has migrated to LevelDB. SpiderByte uses Berkeley DB to store the blockchain index, the wallet, peer addresses and other information. LevelDB is reportedly:
  1. Quite buggy
  2. Project abandoned about 1 year ago
  3. More prone to file corruption than Berkeley DB
2. **Simple structures for speeding process:** where necessary, and following the growth of the network, simple improvements to keep the software going at the right pace, like the drastically reduced bulk operation at boot time, optimizations around the code and smooth network operation.
3. **Maintaining backward compatibility:** since the protocol is quite well defined and stable, there's no need to re-invent the wheel. Backward compatibility is of the utmost importance, because no developer, no matter how good, is infallible. A small bug could compromise the functionality and create a dangerous network split. We will strive with all forces to guarantee no network splits.
4. **Be ready to upgrade:** by choosing strong, portable bases (Berkeley DB, QT Framework), we are ready to go to the next step when used versions become obsolete and is time to upgrade.
5. **Keep the tools for repair:** SpiderByte wallet is one of the few that still keeps the wallet Check/Repair/ZAP tools. These tools are extremely helpful when problems arise with the wallet (like a sudden power outage or system hanging, leading to abrupt shutdowns).

For the future development, that will take SpiderByte software into years to come:

1. **Dev base:** keep a solid and stable development base (2-3 fixed programmers), with occasional help coming from the community.
2. **Strong support:** first hand communication as much as possible between developer and end user, to overcome most difficulties and debug most situations, giving the best experience possible (**never** let the user feel abandoned !)
3. **Always looking for new technologies:** *"there are always people out there who make the wheel rounder than you"*, and when they do, we will be ready to take advantage of that, with final user experience always as top priority.

## 12. Files structure

The data folder for the SpiderByte folder varies from one operating system to another, but the data structure herein is the same, here below is an example of a data folder's content (Linux OS):

Name	Size	Modified	
database	9 items	12:50	☆
themes	109 items	19 Feb 2020	☆
blk0001.dat	2.1 GB	17 Jan 2022	☆
blk0002.dat	1.8 GB	12:50	☆
blkindex.dat	3.5 GB	12:50	☆
db.log	7.3 kB	17 Jan 2022	☆
debug.log	8.5 MB	12:50	☆
peers.dat	854.0 kB	12:49	☆
rules.dat	2.2 kB	Yesterday	☆
SpiderByte.conf	281 bytes	9 Mar 2022	☆
txindex.dat	1.6 GB	12:50	☆
wallet.dat	60.9 MB	12:50	☆
.lock	0 bytes	17 Jan 2022	☆

- **./database:** contains the log files for the transactional Berkeley DB
- **./themes:** may or may not be present, depending on your OS (may be elsewhere) and contain the color/font themes of your wallet software.
- **bindex0001.dat:** this is the boot index file. Not critical. If corrupted, simply delete this file and restart the wallet. The wallet will recognize that the file is missing and it will re-index the blockchain.
- **blk0001.dat:** the blockchain file. Important but not critical, can be rebuild by a resync if corrupted or lost.
- **blkindex.dat:** the blockchain Berkeley DB index file. Goes in pair with the previous one.
- **db.log:** the DB error/info log file
- **debug.log:** the wallet DB error/info log file. May be useful to developers when encountering issues.
- **SpiderByte.conf:** may be present or not, depending on your installation. Normally is not necessary, unless you are an advanced user.
- **peers.dat:** contains the addresses of known peers. If the file is corrupted or missing, it will just be rebuilt at the next boot.
- **→ wallet.dat ←** : this is the most important file of them all. It contains all your keys and public keys for receiving and mapping your coins. If you loose this file, you will loose all your coins. Backup this file frequently. More advanced users can also make a paper-wallet using the command line in the console (Help/Debug/Console).

## X. MASTERNODES

Masternodes are used in special two-tier type currency, in which the layer of who-do-what is subdivided into two, and essentially the govern the background protocol of the network. For example, DASH is a two-tier network using Masternodes:

### *“Masternodes*

*Unlike Bitcoin's single-tier network of miners, Dash utilizes a two-tier network of masternodes and miners.*

*As in Bitcoin, Dash miners secure the network by providing proof of work. The second tier of the Dash network consists of masternodes, which perform PrivateSend, InstantSend, and governance functions. In future releases aimed at improving ease-of-use, the masternode network will store encrypted data relating to user and merchant accounts (DashDrive), and will enable third party clients to interact with the Dash network via a decentralised API (DAPI).” (ref [https://en.wikipedia.org/wiki/Dash\\_\(cryptocurrency\)#Masternodes](https://en.wikipedia.org/wiki/Dash_(cryptocurrency)#Masternodes))*

SpiderByte is a single-tier network, and does **not** plan in the foreseeable future to become a two tier network, thus adding Masternodes. The well balanced combined capacity of PoS and PoW will suffice to make the network stable in the long run.