```
This document contains a series of "shioncoin internet protocols" (SIP) which
describe the behavior of the shioncoin virtual currency computer software program.
```

Table of Contents

Introduction

The official coin ticker symbol of this software is "SHC". The ShionCoin (SHC) Virtual Currency is Scrypt based (LTC compatible). The public source code of the software is written in a combination of the C and C++ programming language. A command-line utility is provided in order to access the ShionCoin (SHC) currency service.

The ShionCoin virtual currency is compatible with the generic "scrypt coin service" capabilities. All BIP protocol specifications implemented by the litecoin virtual currency are candidates as standards for use by the ShionCoin virtual currency service. Several newer BIP standards, including BIP34, BIP65, BIP66, BIP68, and BIP141, have been implemented. While protocol specification compatibility is emphasized, the contents of shioncoin extended transaction structures are not compatible with other scrypt-based currency blockchains.

The majority of the disk databases utilized by the ShionCoin (SHC) virtual currency are in form of a "mapped file". A mapped file is buffered in the operating system's own disk buffer for expediency.

As of 2019, the block-chain has around 100,000 blocks generated. The memory foot-print of the virtual currency service is currently around one-hundred megabytes (100M) of memory, and grows to around one gigabyte (1G) at about 2 million blocks.

A maximum of half a billion coins will be generated. The reward mechanism is designed to provide a block generation incentive over a prolonged period of time.

Extended Transactions

The underlying block-chain supports "extended transactions". These transactions store both auxiliary and internal block-chain processing information.

The "Matrix Transaction" extended transaction is used in order to encapsulate a verification matrix and a spring matrix. The verification matrix is a 3x3 grid which sums checksums of block hashes periodically on the block-chain. This matrix is generated, as needed, when a new block is formed in order to supply an additional layer of security for the block-chain's integrity.

The spring matrix contains a pre-calculated bitvector map of over one million publicly accessible locations. Creating a spring matrix transaction which references a location that has not been claimed yet results in a single SHC coin reward. This reward is deducted from the miner's reward fee.

The "Alias Transaction" is used in order to create labels which reference a coin address. Aliased coin addresses provide a means to supply a public name for transaction operations. You can specify an alias name with the command-line coin by prefixing a "@" symbol (i.e. "wallet.send account-name @alias-name 1").

The "Alt Chain" transaction is used to manage the ShionCoin colored alternate block-chains. Each individual chain is identified by a 128-bit color code. The block difficulty is arbitrarily low, by default. in order to provide a method to submit transactions as a full block. Block-chain parameters, such as the difficulty, minimum transaction fee, and block rewards can be configured when a new chain is generated.

The "Context Transaction" is used to store auxiliary information. This information may be associated with a person, place, or time-frame. The name of the context is hashed; meaning you must know the name before-hand in order to retrieve it. Context transactions expire after two years after being created or updated. The context name typically includes a colon-ized prefix (i.e. "url: http://domain.com") and the data content is typically in JSON format.

A simple identification transaction can be used in order to create a "certified transaction" for coin transfer, and more complicated certificate chains can also be established. The "simple identification" can also be used to tag a location on the block-chain, and is how locations in the spring matrix are claimed.

The "Certificate Transaction" is used in order to create a certified chain of transactions. The protocol used is similar to X509, and can be exported as a X509 certificate.

An "Asset Transaction" is also derived from a certificate, and is intended to constitute some sort of authoritative ownership over an undisclosed concept or property.

A "License Transaction" is derived from a certificate, and is intended to constitute some sort of authoritative licensing grant to authorize some action. For example, the light DRM supplied in the libshare software suite is compatible with licenses generated on the ShionCoin (SHC) block-chain.

An "Exec Transaction" is used in order to publish and run methods against SEXE

classes. Publishing and processing SEXE classes provides a method to create smart contracts which, among other things, are capable of managing tokens and event registration. The SEXE source code is based on LUA, and is provided as part of the the libshare software suite.

The "Offer Transaction" is used in order to transfer coins between one virtual currency block-chain and another. For example, an individual offers to exchange two colored alt-coins for one SHC coin. Another individual can then accept the offer with an exchange between block-chains ultimately occurring. The offer transaction utilizes a OP_CHECKALTPROOF opcode as part of the transaction signature in order to ensure that the exchange occurs safely.

SIP 1: Extended Transactions

```
Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2018-10-10
```

Abstract

This SIP describes a new "extended" transaction type for
the ShionCoin scripting system, and defines additional validation
rules that apply only to the new transactions.

Motivation

The purpose of extended transactions is to embed auxiliary
information into a transaction as a store of information.

Specification

A extended transaction type that is relayed and included in mined
blocks is defined as:

Format: [EXTMODE] [EXTOP] OP_HASH160 [EXTHASH] OP_2DROP
The EXTMODE is a OP_EXT_XXXX script mode as defined below.
The EXTOP is a extended transaction operation type as defined below.
The EXTHASH is a SHA256+RIPEMD160 hash of the extended transaction data.

This script is intended to be used as a prefix and if no additional
script is included then a OP_RETURN is appended.

Extended transactions can be chains be including a previous extended operation
transaction as an input for a new transaction.

The contents of an extended transaction's data layer is not
verified for integrity in order to be allowed onto the block-chain.

Operational Modes

All extended transaction operations use the same pool of modes:

```
OP_EXT_NEW  0xf0
OP_EXT_ACTIVATE  0xf1
OP_EXT_UPDATE  0xf2
OP_EXT_REMOVE  0xf3
OP_EXT_GENERATE  0xf4
OP_EXT_TRANSFER  0xf5
OP_EXT_PAY  0xf6
OP_EXT_VALIDATE  0xf7
```

Extended Operations

The extended operation codes range from interpreted as small integers with a range of 0 - 15.

Here is a list of the established operations:

```
OP_ALTCHAIN  0x04
        Used to deploy SIP22 (colored alternate blockchain).
OP_CONTEXT  0x05
        Used to deploy SIP10 (auxiliary context).
OP_EXEC  0x06
        Used to deploy SIP26 (executable SEXE classes).
OP_MATRIX  0x09
        Used to deploy SIP20 and SIP21 (verification and spring matrix).
OP_ALIAS  0x0a
        Used to deploy SIP6 (public coin address aliases)
OP_OFFER  0x0b
        Used to deploy SIP11 (coin exchange offer)
OP_IDENT  0x0c
        Used to deploy SIP8 and SIP21 (identification stamps)
OP_CERT  0x0d
        Used to deploy SIP5 (certification).
OP_LICENSE  0x0e
        Used to deploy SIP24 (certified license).
OP_ASSET  0x0f
        Used to deploy SIP25 (certified asset).
```

Typical Usage

Extended transactions typically occur as a series of transactions complimenting the underlying extended transaction modes.

The first transaction will fund the operation, the subsequent transactions will use the input of the preceding transaction to create a chain, and finally a terminating transaction will release the funds as a miner transaction fee. Please keep in mind this is a general guideline and is not intended to be implied as a requirement.

Here is an example of a small chain of extended transactions:

An extended operation is funded from a regular account.

```
TX {
  hash: 1,
  vout:{
    script: OP_EXT_NEW [EXTMODE] OP_HASH160 [EXTHASH] OP_2DROP [EXTADDR],
    nValue: 1.0
  }
}
```

The preceding extended operation transaction is used as an input to create the subsequent transactions. Additional "regular account" funding may be necessary.

```
TX {
  hash: 2,
  vin: { hash: 1},
  vout: {
    script: OP_EXT_UPDATE [EXTMODE] OP_HASH160 [EXTHASH] OP_2DROP [EXTADDR],
    nValue: 0.9998
  }
}
```

The preceding extended operation transaction is used as an input to create the subsequent transactions. Additional "regular account" funding may be necessary.

```
TX {
  hash: 3,
  vin: { hash: 2},
  vout: {
    script: OP_EXT_REMOVE [EXTMODE] OP_HASH160 [EXTHASH] OP_2DROP OP_RETURN
    nValue: 0.0001
  }
}
```

The reasoning here is to provide a clear chain of events that can be identified on the block-chain.

The terminating operation, if there is one, sends the remainder of the operational funds to the originating account, burns the coins, and/or submits the funds as a transaction fee in the form of a miner reward.

In most cases this ensures that the operation can be linked to a subsequent extended operation -- thereby solidifying the intent of termination. Similar to regular fund transfer transactions, extended operations are capable of being "rolled back" in the event of a chain re-organization.

Backwards Compatibility

Extended transactions can still be decoded by a traditional scrypt coin service by not unserializing the extended transaction data layer. The script sequence used is designed to be compatible with common scrypt based protocols.

SIP 2: Matrix Transaction

```
  Layer: Core
  Author: Neo Natura < support@neo-natura.com >
  Status: Final
  Type: Standards Track
  Created: 2018-12-29
```

Abstract

This SIP defines a new structure called a "matrix" that is committed to blocks as an extended transaction.

Motivation

The matrix extended transaction is used to store a set of numbers computed with a arbitrary algorithm.

Implementation

The matrix extended transaction uses the core extended transaction structure with the addition of the following variables:

```
nType (uint32)
  The type of matrix.

nHeight (uint32)
  An associated block height.

hRef (uint16)
  Reserved.

vData (uint32[3][3])
  The payload contents of the matrix.
```

SIP 3: Transaction Certification

```
  Layer: Core
  Author: Neo Natura < support@neo-natura.com >
  Status: Final
  Type: Standards Track
  Created: 2018-12-29
```

Abstract

This SIP defines a new structure called a "certificate" that is committed to blocks as an extended transaction.

The certificate extended transaction provides a means to certify credentials.

Motivation

The certification of auxiliary information provides a method to track identity association and provide credentials to third-party software.

Specifications

The Certification of a transaction is performed by creating chains of extended transactions which utilize an "Ident" structure or a "Certificate" structure (see below).

Several other extended transactions utilize the "Ident" or "Certificate" extended transaction structure as a base.

Structure

The "Ident" extended structure consists of the following:

```
{
                /* A geodetic location. */
                shgeo_t geo;
                /* A coin address key. */
                cbuff vAddr;
                /* A mode indicator. */
                unsigned int nType;
}
```

The "Certificate" extended structure is a extension of the "Ident" structure which includes the following;

```
{
                /* A reference to a parent record. */
                uint160 hashIssuer;
                /* A signature generated from the underlying transaction. */
                CSign signature;
                /* A free-form data pay-load. */
                cbuff vContext;
                /* The fee charged to derive this certificate. */
                int64 nFee;
                /* An attribute modifier. */
                int nFlag;
}
```

Usage

An "Ident" by itself can be embedded into a "transfer funds" operation as part of a certified send or by "stamping" some auxiliary information.

The "Certificate" structure is used by certificates, licenses, and assets. Typically, a chain authority certificate is generated and then derived certificates are assigned for auxiliary
purposes.

A license differs, in that, it can be derived from a suitable certificate, with a optional fee, without the consent of the owner.

An asset is derived from a certificate, and provides a 4k payload of information referencing a tangible asset.

SIP 4: Alternate Blockchain Transaction

Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2018-12-29

Abstract

This SIP defines a new transaction which is capable of tracking an alternate block-chain.

The alternate blockchain extended transaction stores the block header and the individual transactions associated.

Motivation

By providing a method to track an alternate blockchain it becomes possible to create "colored" chains which can have independent value, purpose, and blockchain parameters.

Implementation

The alternate blockchain extended transaction is built on top of the standard extended transaction structure.

A 32-bit color code identifies each individual block-chain.

Each altchain transaction holds a block header and a set of transactions.

A standard block header structure is used:

```
{
  unsigned int nFlag;
  uint256 hashPrevBlock;
  uint256 hashMerkleRoot;
  unsigned int nTime;
  unsigned int nBits;
  unsigned int nNonce;
}
```

Followed by one or more transactions with the following structure:

```
{
  unsigned int nFlag;
  std::vector vin;
  std::vector vout;
  unsigned int nLockTime;
  cbuff vchAux;
}
```

The "vchAux" variable is a proprietary packet of information that is optionally defined and used by the developer of the particular alternate blockchain.

Segregated witness operations are not supported for the alternate blockchain extended transaction.

All coinbase transactions include a BIP39 input which encapsulates the block height of the block being generated.

New Chain

A new chain is created using the OP_EXT_NEW opcode.

The coinbase consists of an "OP_RETURN 0x0" output script with an output coin value determined by the underlying "reward value" blockchain parameter provided.

Parameters are appended after the "OP_RETURN 0x0" is the format "< mode > < value >" where < mode > is a 8-bit number representing the parameter and < value > consisting of a 8-bit code.

The blockchain parameters provided are beyond the scope of this document.

Update Chain

Supplemental blocks are appended to the chain, for the particular color specified, using the OP_EXT_UPDATE opcode.

SIP 5: Certificate Transaction

Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2019-01-26

Abstract

This SIP defines a new transaction called the "Certificate Extended Transaction".

This extended transaction is used in order to generate certification certificates which contain digital signatures referencing their previous chain (parent) transaction. A root-level certificate will have no previous certificate in the chain.

Specifications

A certificate contains a title of up to 135 characters, an coin address of the creator, an optional parent chain certificate hash, and an optional fee.

Certificates cost about 6.5 SHC and go down in cost over time.

A certificate will include the geodetic location of the node which generated the transaction unless the node is configured otherwise.

A certificate is signed against the creator's coin address. A certificate may be flagged to allow derivatives to be created from itself.

Reserved Use

The certificate extended transaction is modeled in a manner as to be compatible with generic digital certificate chains.

Additional extensions may be provided at a later date which affect the version, type, and flag variables.

SIP 6: Alias Transaction

Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Informational
Created: 2018-12-29

Abstract

This document introduces the ability to associate a string based label which

references a ShionCoin coin address.

Motivation

Having a string based alias for a ShionCoin coin address provides a means to supply a human-digestible label to what is otherwise an arbitrary code.

Public entities, such as a store-front, can assign a label to an address in order to broadcast a receiving address and/or provide other facilities that can be associated with a coin address.

Usage

The coin alias functionality is utilized by a shioncoin node by providing a means to replace references to a coin address with a alias name prepended by an ampersand (@).

Specification

The alias extended transaction structure is an extension of the "Ident" certificate structure. The alias transaction provides a label of up to 135 characters which references a ShionCoin coin address.

The alias is defined as having a "type" which indicates what type of coin address is being referenced. An coin address alias can be created, update, and removed.

Each alias has a life-span of 12 years past when it was created or the last time it was updated.

Examples

For example, sending funds to another address:
wallet.send bank @AliasName 5.5

SIP 7: Executable

Layer: Core
Title: Executable (Extended Transaction)
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2018-12-29

Abstract

This proposal introduces the ability to create executable calls based on applications published to the ShionCoin block-chain. See SIP26 for a more comprehensive guide on the how a ShionCoin node processes application calls.

Motivation

The Executable Extended Transaction provides the base structure for publishing and executing calls.

Specification

```
The executable extended transaction is composed of the following structure:
{
    /* the originating address of the transaction. */
    uint160 kSender;
    /* the executable code of a SEXE class. */
    cbuff vContext;
    /* a signature certifying the primary contents of the exec/call. */
    CSign signature;
    /* the system time at which the call was initiated. */
    int64 nTime;
    /* the blockchain height at which the call was initiated. */
    int64 nHeight;
    /* the coin value sent to process the method call. */
    int64 nValue;
    /* a hash of the result returned from the method call. */
    uint64 hResult;
    /* the SEXE executable class. */
    uint160 hExec;
    /* the previous call hash */
    uint160 hPrev;
    /* a checksum hash of the class's user-data. */
    uint256 hData;
    /* the name of the method called. */
    cbuff vchMethod;
    /* the arguments passed into the class method. */
    vector vArg;
    /* a list of transactions generated by this exec call. */
    vector vTx;
}

An executable transaction is composed of either a OP_EXT_NEW or a OP_EXT_GENERATE
operation. A OP_EXT_NEW operation publishes a new set of executable code to the
ShionCoin block-chain.

The executable code may be in text or binary format. Published executable code will
expire after 48 years.

An OP_EXT_GENERATE operation registers that a call was performed into a particular
application.

The maximum size of the executable code payload is 780,000 bytes.
```

SIP 8: Ident

```
  Layer: Core
  Author: Neo Natura < support@neo-natura.com >
  Status: Final
  Type: Standards Track
  Created: 2018-12-29
```

Abstract

This SIP defines a new "Ident" extended transaction. The ident extended transaction is used in order to certify the originating sender of funds or other auxiliary

information.

Motivation

The ident extended transaction is intended to provide a method for third-party application to verify the originating identify of funds being sent.

In addition, the transaction may be used in order to "stamp" auxiliary information onto the block-chain. This can include general information or something more specialized like a geodetic address.

Specifications

The Certification of a transaction is performed by creating chains of extended transactions which utilize an "Ident" structure or a "Certificate" structure (see below). Several other extended transactions utilize the "Ident" or "Certificate" extended transaction structure as a base.

Ident Structure

The "Ident" extended structure consists of the following:

```
{
                /* A geodetic location. */
                shgeo_t geo;
                /* A coin address key. */
                cbuff vAddr;
                /* A mode indicator. */
                unsigned int nType;
}
```

Usage

A "Certified Funds Transfer" is comprised of a regular fund send transaction with the addition of a "Ident" extended transaction.

The ident extended transaction optionally tracks the geodetic location of the originating node server, and a payload of textual information up to 135 characters.

An ident transaction can specify a geodetic location by using the format "geo:latitude,longitude" where latitude and longitude are in decimal degrees format.

See SIP21 for additional usage of ident extended transactions utilizing geodetic locations.

SIP 10: Coin Address Alias

```
  Layer: Core
  Title: Coin Address Alias (Extended Transaction)
  Author: Neo Natura < support@neo-natura.com >
  Status: Final
  Type: Standard Track
  Created: 2018-12-29
```

Abstract

This document introduces the ability to associate a coin address with a common label.

Motivation

The Alias Extended Transaction provides the ability to create an alias between a label (textual string) and a virtual currency coin address destination.

Specification

The alias extended transaction is used in order to register and update public coin address aliases. The alias extended transaction is composed of three different parts. A string based label with a maximum of 135 characters, a variable sized coin address, and the type of coin address being referenced.

The current proposed implementation includes a single type which indicates a coin address in it's final string based representation.

In other words, no internal information about the address is retained with this type that the coin address is composed of, such as a public key. This type supports pubkey compressed, witness, bech32, and dilithium address formats. A new alias is created using the OP_EXT_ACTIVATE opcode.

An alias is updated using the OP_EXT_UPDATE opcode. An alias may be removed with the OP_EXT_REMOVE. Only the original creator may perform a valid OP_EXT_UPDATE or OP_EXT_REMOVE operation.

A context will expire after 12 years. Subsequent updates reset this time-span, and provide a means to retain ownership past the initial lifetime.

Transaction Fee

The transaction coin fee of creating or updating a context can be derived via the following formula where nHeight is the associated block height:

```
{
        double base = ((nHeight+1) / 10240) + 1;
        double nRes = 5000 / base * COIN;
        double nDif = 4750 /base * COIN;
        int64 nFee = (int64)(nRes - nDif);
        nFee = MAX(MIN_TX_FEE(iface), nFee);
        nFee = MIN(MAX_TX_FEE(iface), nFee);
        return (nFee);
}
```

Client Usage

The RPC console program will interpret all coin destinations which are prefixed with a ampersand as an alias.

For example: shc wallet.send "" @neonatura 0.1

Individual client interfaces may deploy custom methods of supplying public alias references.

SIP 11: Coin Exchange Offer

Layer: Core
Title: Coin Exchange Offer
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2019-01-27

Abstract

This SIP describes a new Offer Extended transaction, and how it is used in order to transfer funds between two different block-chains.

Motivation

The purpose of the offer extended transaction is to provide a method perform a decentralized exchange of coins from one block-chain to another. This functionality is currently only compatible with the exchange of SHC coins with the ShionCoin Alternate Colored Block-Chain coins.

Specification

The Offer Operational Fee, the cost to publish offer transactions, is double the minimum transaction fee (0.0002 SHC). Some transactions may retain a minimum transaction fee, in addition to the offer operational fee, in order to perform a additional action later on in the process. An offer extended transaction will expire after 24 minutes.

A new offer is published on the block-chain using a OP_EXT_NEW operative. The new offer will include a associated number of SHC coins, the color hash of the alt-blockchain coins being converted, the rate of the conversion, and the SHC value range that would be accepted. An additional "Offer Operation Fee" is also required. A secondary party may then perform a OP_EXT_ACTIVATE operative in order to accept the offer.

The OP_EXT_ACTIVATE extended transaction is sent to a null address (OP_RETURN), and a separate transaction is created on the colored alternate block-chain whereby the value of the color coins in sent to one-self for holding.

The original party can perform a OP_EXT_GENERATE against an accept offer in order to complete the exchange. When this occurs, a transaction is sent on the ShionCoin blockchain with a IF-ELSE script. This conditional script will send coins to the acceptee if a pre-determined transaction hash exists on the colored block-chain, and otherwise will return the coins once the generate offer transaction has expired.

The OP_EXT_GENERATE script is composed of the following:
OP_EXT_GENERATE OP_OFFER OP_HASH160 offer_hash OP_2_DROP alt_hash color_hash OP_CHECKALTPROOF OP_IF dest_addr OP_CHECKSIG OP_ELSE wait_time OP_CHECKLOCKTIMEVERIFY OP_DROP return_addr OP_CHECKSIG OP_ENDIF

The "offer_hash" is the hash of the OP_EXT_GENERATE's offer structure, the "alt_hash" is the tx hash on the colored alt-chain which sends coins from the acceptee to the originator, the "color_hash" is the hash representing the

color of the alt-chain, the "dest_addr" is the acceptee's coin address on the ShionCoin block-chain, the "wait_time" is the time before the generate transaction expires, and the "return_addr" is the originator's own address.

The node which originated the acceptee's OP_EXT_ACTIVATE offer transaction will automatically send the colored alt-coin funds to the originator, once a OP_EXT_GENERATE is processed, using pre-determined TX contents. The hash for this TX is included as part of the activate offer transaction.

The original sender or the acceptee can cancel the exchange using a OP_EXT_REMOVE transaction. This can be used by the originator or acceptee before a OP_EXT_GENERATE transaction is published, or after a set of transactions has expired due to inaction.

SIP 12: Dynamic Blockchain Parameters

Layer: Core
Author: Neo Natura < support@neo-natura.com >

Status: Extensionable
Type: Standards Track
Created: 2018-12-15

Abstract

This SIP defines a new extended transaction called a "parameter" that can be attached to shioncoin blockchain transactions. The structure defines a set of rules that are being proposed for the blockchain.

Motivation

The "extended parameter transaction" provides the ability to dynamically change aspects of the blockchain based on a user consensus. In addition to providing the participants of the network control over the blockchain's behavior, this mechanism provides a method to alter behavior in order to mitigate behavior that is later deemed inappropriate.

In other words, by providing a dynamic method of altering blockchain parameters, a mechanism is provided to apply conditions to the current state of things as opposed to what was considered suitable when the functionality was originally implemented.

Specification

The Param extended transaction is being introduced by bit 6 of the VersionBits mechanism. The starting deployment date is 01/01/21 and the timeout is 01/01/22. The testnet is set to one year earlier.

The embedded param ext-tx specifies a preferred value for a single parameter mode. The output coin value of the param ext-tx is always 0 coins.

The parameter value must be either half or double the current established reference value. For example, since the default maximum blocksize of the shioncoin blockchain is 4megs, the "max blocksize" param mode can initially specify a value of "2048000" or "8192000".

To include a parameter specification, an additional output is added and the "TXF_PARAM" flag is enabled for the transaction.

The param output has a traditional extended transaction format:
OP_EXT_UPDATE << OP_N(OP_PARAM) << OP_HASH160 << hashParam << OP_2DROP << OP_RETURN << OP_0

A consensus occurs when 10240 or more param ext-txs are submitted and a particular mode value has 90% of the votes. Each param ext-tx will expire after 30 days. For example, if at last 10240 param outputs are generated in less than 30 days, and at least 90% of them specify a maximum block size of 8192000, then the new blockchain parameter will be applied once the 90% mark is hit.

The rpc command "sys.info" provides a method to review what the current parameter values are. The rpc command "param.list" returns a list of all params that are not expired.

Parameters

This SIP introduces two initial parameters; the "maximum block size" and the "minimum relay fee". The proposal supplies a method to insert a param ext-tx into a transaction. The transaction may be coinbase transaction, a regular coin transfer transaction, or another extended transaction.

Maximum Blocksize

The maximum blocksize denotes the maximum non-witness byte size of a block committed to the shioncoin blockchain. The shioncoin blockchain, by default, has a maximum block size of 4096000 bytes. By supplying a dynamic method of adjusting the amount of information contained in each block, the participants of the shioncoin blockchain network can decide, by a consensus decision, how much information is prudent to be stored in a single block.

Minimum Fee Rate

The minimum fee rate param references the "minimum transaction relay fee". This is the absolute minimum that a transaction may have and still be accepted into a node's transaction mempool. This value is not to be confused with the "minimum transaction fee" which denotes a multiplier value against each kilo-byte of information in the transaction.

The shioncoin blockchain, by default, has a 0.00001 minimum transaction relay fee. Similar to aspects of the maximum block size, this parameter affects the over-all cost of sending transactions on the shioncoin blockchain.

Extensions

A particular extension could affect previous node behavior (hard fork), and it is the intent that if this occurs the change be introduced in a manner that mitigates this to only affecting very old nodes.

For example, by introducing functionality in a particular version, and then waiting until a later version to introduce the param which most nodes should already be "ready" for. If possible, though, any introduction of new dynamic blockchain parameters would be preferred to be entirely backwards compatible.

SIP 20: The Validation Matrix

Layer: Core
Title: The Validation Matrix
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2018-10-13

Abstract

This SIP describes the specific usage of the Matrix extended transaction type for the purposes of blockchain validation with the ShionCoin scripting system.

Motivation

The purpose of the Validation Matrix is two-fold; one purpose is to provide an additional redundant mechanism to ensure the blockchain's validity, and second to supply a method for a consensus of nodes to generate a dynamic checkpoint on the blockchain.

Specification

Every 27 blocks a "validation matrix" is required to be generated on the block-chain in the case where there are no other transactions in the block being submitted other than the coinbase. The matrix is encapsulated in the coinbase using a matrix extended transaction, and has a coin output of one coin. The other-wise total reward of the coinbase is reduced by a single coin.

The numeric information in the "validation matrix" is composed of a running hash split into 9 cells which is directly based on the preceding block-chain. The validation matrix acts as a redundant security measure in order to ensure that the block-chain is viewed in a consistent manner between all nodes.

Note that the validation matrix does not provide in itself anything more than an additional checksum mechanism to the existing block-chain.

In addition to providing a supplemental level of integrity to the block-chain in the form of a "redundant merkle hash", the validation matrix is also used to initiate a sequence of events that potentially lead to a new dynamic checkpoint being established. This is performed by a consensus vote led by active participants (referred to as notaries) on the block-chain. Each node which generates a "validation matrix" becomes a candidate to become a "notary".

When enough notaries are available the validation matrix which use a multi-sig output which contains the pubkeys from all participating notary nodes.

A "notary transaction" has a single input; the latest validation matrix transaction. It is signed by each node (in order) until the transaction is finalized and complete. Once, and if, completed, a dynamic checkpoint will be made for the block which contained the "validation matrix" transaction.

Note that only the last validation matrix block on the block-chain will be considered as a potential checkpoint.

The output of the notary transaction is a "null return" which will result in the original coin being transformed back into a claimable miner transaction fee (less a

small amount for additional ensued transaction fees).

Dynamic Checkpoints

Dynamic checkpoints, in essence, are similar to the delayed proof-of-work (dPOW) concept.

The dynamic checkpoints are intended as a way to provide resilience, and therefore extra security, for the integrity of the block-chain. By providing additional "agreed upon" checkpoints dynamically it becomes more difficult to maliciously alter or otherwise by performing a "51% attack".

The output of a validation matrix coinbase transaction can either be OP_RETURN or a multi-sig redeem script which includes a minimum consensus of notary pubkey addresses.

Here is an example of a notary transaction:

{

"txid":"f51d4c04a6ab1483a6562960140d1eee83a0f8d7ff91b4e68375c952c1a5bcb0",
"vin":
[{"txid":"2788f12789b27c9167b56a34c696dc117bb5939d00b70a3474aae451cbed3962","vout":0,"scriptSig":"0
3045022100bba304fb584f1ace2f101fd7ada1d36475da8f3ad9178e99029a020614e3208202204e9ae644ea6a6d5d184097f4ce4ba6a17cf92201936aa37d15637f8ee32d0fd601","sequence":4294967295}],
"vout":
[{"value":0.00000000,"n":0,"scriptpubkey":"OP_RETURN","reqSigs":1,"type":"return"}]

}

An incomplete "notary transaction" is submitted and each node with a public signature included in the transaction's multi-sig redeem script will sign the transaction, and then submit it again. This is done in the order of the pubkeys listed in the multi-sig redeem script.

At no point will a incomplete "notary transaction" be added to the transaction memory pool and the "input sequence" to a non-final value until all signatures have been squired. Nodes which do not have a public key address listed in the multi-sig redeem script will still relay the transaction even if it's incomplete.

In order to avoid redundancy, a node will only sign the "notary transaction" if it's own public is the next in line that has not been signed pertaining to the multi-sig pubkey order.

At this point, a new checkpoint will be established for the block containing the validate matrix transaction providing the following three criteria are met:

1. The "notary tx" references the latest Validation Matrix established,
2. the "notary tx" has a single input containing a Validation Matrix transaction, and
3. the "notary tx" has a single output OP_RETURN and a coin value of zero (0).

SIP 21: The Spring Matrix

Layer: Core
Title: The Spring Matrix
Author: Neo Natura < support@neo-natura.com >

Status: Final
Type: Standards Track
Created: 2018-12-29

Abstract

This SIP describes a pre-compiled bitvector matrix which encapsulates a set of locations, and how to access it's components. The Spring Matrix is composed of over one million public latitude/longitude geodetic locations. These locations are stored as bitvectors in a set of cells comprising the matrix. All operations against the matrix consist of removing bitvector values, and therefore no new information will ever be generated.

Motivation

The purpose of the Spring Matrix is to provide a low yielding reward for the identification of a particular geodetic location currently present in the Spring Matrix. All of the locations in the Spring Matrix have been predefined. Claiming a location will remove the bitvector values associated with it's latitude and longitude permanently from the Spring Matrix. The degree of hardness naturally increases as a smaller pool of places become available to claim. A single coin is rewarded for successfully identifying a location in the Spring Matrix. This value is arbitrarily low in order to avoid interfering with the economics of the mining ecosystem.

Specification

The act of claiming a reward is performed by generating a "identification stamp". The location being claimed has no relevance to the physical location of the user claiming it, although additional constraints can be applied (i.e. for VR / geocaching applications) if desired.

When a location is "claimed" by an identification stamp transaction, the underlying bitvector value is removed from the Spring Matrix.

The removal of a bitvector will prevent the same location from being claimed multiple times, and may also prohibit other locations that rely on aspects of the bitvector from being claimed as well.

When a identification transaction is received containing a particular geodetic location, and that geodetic location is currently present in the Spring Matrix, then the next block will deduct a single coin from the block reward and generate a supplemental transaction containing a single coin output to the coin address specified in the identification transaction.

SIP 22: Alternate Blockchain

Layer: Core
Title: Alternate Blockchain (Extended Transaction)
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2018-10-07

Abstract

The Colored Alternate Blockchain provides a method of providing supplemental block-chains embedded into the main ShionCoin blockchain which can have their own custom parameters.

Each alternate blockchain is assigned a unique 128-bit RGBA color code derived from the label of the chain.

Sending funds and generating new blocks are mined in a similar fashion to regular blocks, but have a reduced minimum difficulty in order to accommodate CPU mining.

Motivation

The colored alternate block-chain is intended to provide a sub-set of individual chained blocks which can be used for an alternate purpose or capability then the primary ShionCoin Blockchain.

Implementation

Each colored block-chain has a unique genesis block and a subsequent set of chained blocks. The blocks contain transactions in a typical fashion and structure. The transactions may embed an additional payload, depending on their version, which can be used for extension or auxiliary purposes. When a chain is created it can have custom parameters applied to it in relation to how it behaves. See "Blockchain Parameters" below for more information.

The "alternate block" structure is composed of the typical "version", "previous block hash", "merkle root hash", "time", "difficulty bits", and "nonce". The "alternate transaction structure is composed of the typical "version", "inputs", "outputs", and "lock time". In addition, the alternate transaction structure will serialize an additional unsigned character vector when the transaction version is higher than one (1).

The AltChain Extended Transaction, use to encapsulate color alt-chains on the ShionCoin blockchain, is composed of a flag bitvector, the hash of the colored chain, a alternate block structure, and a set of alternate transactions.

A OP_EXT_NEW operative is performed in order to establish a new colored alt-chain. The alternate block will include a single coinbase which has optional blockchain parameters and a single output to null (OP_RETURN) with the initial reward value. The genesis block will always have a difficulty of (~uint256 >> 12). The script for the extended transaction output is "OP_EXT_NEW OP_ALTCHAIN hash OP_2DROP OP_RETURN 0".

Subsequent blocks on the alt-chain use the OP_EXT_UPDATE script operative. The script for the extended transaction output is "OP_EXT_UPDATE OP_ALTCHAIN hash OP_2DROP OP_RETURN 0".

An alternate chain extended transaction fee of 0.001 SHC is applied to both the OP_EXT_NEW and OP_EXT_UPDATE outputs. Since the output is null (OP_RETURN), the SHC coins are burned in the process.

Blockchain Parameters

The parameters for a block-chain are stored in a coinbase output of the genesis block for the alt-chain.

The format of the script is "OP_RETURN OP_0 [ ].." with a prefix of "OP_RETURN OP_0" and a sequence of small number pairs which represent the parameter mode and value.

Here is a table of the parameters and there corresponding values.

01 DIFFICULTY
The minimum mining difficulty to generate a new block.
note: Genesis Block is always >> 12
note: A value less than "4" is recommended for CPU mining.
        X = (~uint256 >> (X+10))
        1 = (~uint256 >> 11)
        2 = (~uint256 >> 12) (etc)
        15 = (~uint256 >> 25)
02 BLOCKTARGET (seconds)
The target time-span in between new block generation.
        1 = 60
        2 = 120 (etc)
        15 = <15 minutes>
03 MATURITY (blocks)
The number of blocks before a coinbase is spendable.
note: The maximum allowed value for this mode is "8".
        1 = 60
        2 = 120 (etc)
        8 = 480
04 REWARDBASE
The base coin value of new block rewards. This is halved based on the REWARDHALF parameter.
note: The maximum allowed value for this mode is "10".
        X = (2 ^ X)
        1 = 1 COIN
        2 = 2 (etc)
        3 = 4
        4 = 8
        5 = 16
        6 = 32
        7 = 64
        8 = 128
        9 = 256
        10 = 512
05 REWARDHALF
The number of blocks before the REWARDBASE is halved.
        X = (1000 * X)
        1 = 1000
        12 = 12000
        15 = 15000
06 TXFEE
The minimum tx-fee to enforce for transactions.
note: The maximum allowed value for this mode is "8".
        X = (10 ^ X)/COIN
        0 = 0
        1 = 10 (0.0000001)
        2 = 100 (0.000001)
        3 = 1000 (0.00001000)

```
4 = 10000 (etc)
5 = 100000 (0.001)
6 = 1000000 (0.01)
7 = 10000000 (0.1)
8 = 100000000 (1)
```

Usage

Blocks generated on the colored alt-chains are created using a CPU miner by the server node. In addition, specialty (specific color) nodes can be created which provide access to stratum facilities. Sending a transaction on an alt-chain typically consists of creating a block along with the transaction. The transactions are still pooled in the event of a chain reorganization.

A transaction will typically provide a funds transfer to another individual, a funds transfer to several individuals, or submitting a data payload.

Transactions containing a data payload are meant to be specific to the particular color of the alt-chain. By proving a custom set of standards to the data content published, a rule set can be enforced on the underlying data permitted to provide a common method of processing the data.

Examples

Here is an example output script of a alternate blockchain extended transaction which contains a genesis block for a colored altchain with block-chain parameters defined as having a difficulty of "~uint256 >> 12" and and block target of 120 blocks:

```
OP_RETURN OP_0 OP_1 OP_2 OP_2 OP_2
```

The "OP_1 OP_2" pair indicates a difficulty of "(~uint256 >> 12)" and the "OP_2 OP_2" pair indicates a block target of 120 seconds.

SIP 23: Certified Coin Transfer

```
Layer: Core
Title: Certified Coin Transfer
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2019-01-27
```

Abstract

This SIP defines a method of sending a coin payment with a verifiable return address.

Motivation

The ident extended transaction is intended to provide a method for third-party application to verify the originating identify of funds being sent.

Specifications

A certified coin transfer is performed in the same manner as a regular coin transfer, except that it includes a reference to a certificate that was created by the sender.

The output script for the payee destination is prefixed with the following:

OP_EXT_PAY OP_N(OP_IDENT) OP_HASH160 [IDENT HASH] OP_2DROP

Usage

A "Certified Funds Transfer" is comprised of a regular fund send transaction with the addition of a "Ident" extended transaction.

SIP 24: Certified License

Layer: Core
Title: Certified License
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track
Created: 2019-01-26

Abstract

This SIP defines a new transaction called a "License". A license is derived from an already established certificate.

Licenses are used in order to indicate a usage of a certificate's context. For example, a software or audio license may be require to be attained before the program will run or be played.

Specifications

A license contains a title of up to 135 characters, an coin address of the creator, and a parent chain certificate hash.

A fee must be paid to generate the license if the certificate being derived specifies one. In addition to any certificate fee, a 0.0001 SHC transaction fee is applied.

A license will include the geodetic location of the node which generated the transaction unless the node is configured otherwise.

A new license is created using the OP_EXT_ACTIVATE extended transaction operation.

SIP 25: Certified Asset

Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Final
Type: Standards Track

Created: 2019-01-26

Abstract

This SIP defines a new transaction called a "Asset". A asset is derived from an already established certificate.

Motivation

The asset extended transaction allows for additional notations to be appended to a certificate. These notations typically reference a code of some arbitrary context. This functionality provides for a method to associate additional attributes to the underlying meaning of the certificate.

Specifications

An asset contains a title of up to 135 characters, an coin address of the creator, a parent chain certificate hash, and a data payload. Up to 135 characters may be used for the payload.

An asset will include the geodetic location of the node which generated the transaction unless the node is configured otherwise. Assets provide a method to attach auxiliary information to a certificate. The asset may itself be a reference (such as a barcode) or contain authority/proprietary information by itself.

An asset will expire when the underlying certificate expires (which is after 48 years).

Asset Fee

Asset transaction initially cost around 8 SHC and goes down over time.

Asset Creation

The extended transaction format is:
OP_EXT_NEW << OP_N(OP_ASSET) << OP_HASH160 << [ASSET HASH] << OP_2DROP

Where [ASSET HASH] is the hash of the underlying asset extended transaction payload.

The certificate being referenced by the asset need only exist, and does not need to be owned by the asset creator.

Asset Update

The extended transaction format is:
OP_EXT_UPDATE << OP_N(OP_ASSET) << OP_HASH160 << [ASSET HASH] << OP_2DROP

Where [ASSET HASH] is the hash of the underlying asset extended transaction payload.

You must be the creator of the asset in order to update it. Only the asset’s payload may be updated.

Asset Removal

The extended transaction format is:
OP_EXT_REMOVE << OP_N(OP_ASSET) << OP_HASH160 << [ASSET HASH] << OP_2DROP

Where [ASSET HASH] is the hash of the underlying asset extended transaction payload.

You must be the creator of the asset in order to remove it.

SIP 26: Executable SEXE Class

Layer: Core
Author: Neo Natura < support@neo-natura.com >
Status: Extendable
Type: Standards Track
Created: 2018-12-15

Abstract

This SIP defines a new Executable Extended Transaction that is used in order to provide smart-contract capabilities. The executable extended transaction consists of publishing a SEXE class and subsequent calls to methods in that class. A persistent data storage mechanism is provided which is synchronized between all nodes.

Motivation

The executable extended transaction provides the capability to create programs which can define rules for a system of operations that deliver consistent results among all participants.

Examples

One of the most popular examples of a smart-contract program is the Token asset. A token tracks an arbitrary metric that can be defined as having a intrinsic value based of the intended purpose of it's use.

Specification

The SEXE programming language is based off of the LUA scripting language. The libshare run-time suite is used in order to provide the development environment.

The executable extended transaction consists of the generation of a new SEXE class on the blockchain or calling a method from a published class.

The SEXE class is provided a permanent data storage area in order to update variables based on the functions called.

The SEXE class can require a SHC coin payment in order to perform an operation. Class method calls that do not require a change to the permanent data storage area do not require a payment.

The minimum fee for a class call which modifies the data storage area is 0.00001 SHC.

The maximum size permitted fora SEXE class published on the shioncoin blockchain is 780000 bytes.

When a class calls is received by a node the basic integrity is validated. The actual execution of the call will only occur on a 'as needed' basis, in that, a user must specifically require the use of a SEXE class before any preceding operations are performed.

Publishing SEXE Class

The extended transaction operation OP_EXT_NEW is used in order to publish a new SEXE class onto the blockchain. The underlying payload containing the LUA code may be in raw text format or compiled [using the “sxc” libshare compiler].

When publishing a new class, the name of the class must be unique. A fee will be charged based on the size of the underlying payload. The executable extended transaction will be signed against the account’s unique execution sender address.

The underlying script used for the executable extended transaction, when creating a new class, has the following format:
OP_EXT_NEW << OP_N(OP_EXEC) << OP_HASH160 << [EXEC HASH] << OP_2DROP << OP_RETURN

The [EXEC HASH] is the has of the underlying CExec strucuture that comprises the payload of the extended transaction (see BIP 7).

Calling a Class Method

A new transaction is generated on the shioncoin blockchain that supplies the method and parameters being called. All nodes will execute the same method and arguments when this class is being referenced. A checksum is supplied in order to ensure that the resulting permanent storage area associated with the class has is consistent on all server nodes.

The extended transaction operation OP_EXT_GENERATE is used in order to indicate a class method is being called, and is comprised of the following format:
OP_EXT_UPDATE << OP_N(OP_EXEC) << OP_HASH160 << [EXEC HASH] << OP_2DROP << OP_RETURN

The node calling the class method must ensure that a high enough fee is being provided or an error may be returned. Execution of class methods that do not require altering the permanent storage area used by the class do not endure any fee, and are not recorded on the shioncoin blockchain.

The SEXE class identifies each caller by the sending coin address that supplies the fee to run the execution. It is therefore necessary to use a unique address per account when executing a class method.

A particular class is not executed on a node until it is needed due to a user attempting to call a class method. Pending execution of method calls on the blockchain, that have not already been performed, are performed in sequence [as they reside on the blockchain] before attempting to execute a new class method call.

SIP 31: BOLO Checkpoint

Layer: Consensus (soft fork)

Author: Neo Natura < support@neo-natura.com >
Status: Proposed
Type: Standards Track
Created: 2018-11-26

Abstract

This SIP describes a method to establish a new dynamic checkpoint on the ShionCoin blockchain by creating a "notarized transaction" on an external blockchain. The external blockchain intended for use is litecoin.

In other words, the "BOLO" system notarizes blocks on the ShionCoin blockchain by creating a special transaction on the litecoin blockchain.

Motivation

A notarized transaction, or a transaction signed by independent users who participate in the ShionCoin ecosystem, provides a method to create a consensus about a particular blockchain and a means to declare it as the official branch. This is only possible if at least 11 participants are in agreement about the current blockchain hierarchy.

The block utilized for a potential checkpoint consensus is the block associated with notarized transaction created by SIP30, and in turn adds an additional layer of security.

Specification

The transactions created on the external blockchain are associated with the following output script.

"Bolo" Notarization Output Script:

OP_RETURN << OP_11 << OP_1 << OP_11 << OP_HASH160 << OP_HASH256 << OP_HASH256 << [HEIGHT] << OP_0

The notarization script references the coin interface (literally Hash160("shc")) as the OP_HASH160.

The first OP_HASH256 is the hash of the SHC block that is being notarized.

The second OP_HASH256 is a merkle hash of the last 1000 blocks prior to the notarized block.

The [HEIGHT] is a 4-byte "integer" of the block's height.

The ShionCoin bolo interface will attempt to notarize any blocks that have a OP_RETURN OP_0" output script suffix and have a coin value equal to or less than 1 coin. This includes both no-output matrix validation coinbase transactions, notarized dynamic checkpoints transactions, or other arbitrary transactions.

Validation matrix dynamic checkpoints are created on the ShionCoin blockchain by several nodes providing a multi-sig transaction verifying a validation matrix.

When a bolo notary transaction references a validation matrix notary tx, it will in

effect provides three layers of security in order to establish blockchain integrity. Refer to SIP30 for more information on dynamic checkpoints generated from validation matrix transactions.

After every 1000 blocks the block-chain is scanned for the first null-output transaction as described above.

Once the block height has reached an offset of 20 blocks each participating node will submit a transaction on the LTC blockchain that can be considered a proposal to generate a notary transaction using the combined outputs.

A bolo litecoin transaction requires that at least eleven (11) nodes to be in agreement with the proposed notary height and it's preceding 1000 blocks of transactions

A notarized bolo transaction will not over-ride a locally established dynamic checkpoint when the bolo transaction references the same or lower height. Participating notaries publish a "bolo proposal tx" on the litecoin block-chain.

The first output contains a 0.00001 coin output value to themselves. A second output is a CScript ID of the final notarization output script with a zero coin value. The potential notary endures tx fee to initiate the transaction, and, for example, with a fee of 0.0001 ltc a total of 0.00011 LTC would be spent for the proposal tx submitted.

A tx which includes all potential notaries is submitted, with a lock time of 20 blocks, until at least eleven participants have provided a signature for the transaction. The transaction will fail after the lock time period if not enough signatures are gathered.

A transaction will not be committed to the blockchain until it's "lock time" has elapsed. A final notarized transaction consists of the described notary proposal outputs (which each contain 0.00001 ltc).

When available notaries with eleven or more signatures has been gathered, the final transaction is committed with the original lock time and a "final sequence" marked for all inputs. A single output is the full notarization output script with a zero coin value.

When the final notarized transaction is committed to the external blockchain, a new dynamic checkpoint will be created on the shioncoin blockchain for the block height of the original SHC block being notarized. The individual inputs comprise the tx fee in order to publish the transaction. For example, a transaction comprising of eleven notaries would generate a 0.00011 LTC paid fee.

The external blockchain transaction is composed of a SIGHASH_ANYONECANPAY input type. This is done in order to allow for "signatures to be gathered" as nodes relay the notary transaction with more compiled inputs. The inputs are sorted in a manner so that the same transaction will be submitted regardless of origin node [where all nodes see the same information].

Non-signed inputs of participants are included until at least eleven signed inputs are acquired. Only active participants which have already committed a notary proposal on the external blockchain are considered for inclusion in the final notarized transaction.

Rationale

This SIP provides a supplemental layer of verification derived from the capabilities provided in SIP30; which itself derives from the verification capabilities provided in SIP20. This intentional "chained" design provides a robust method of verification and integrity.

Backwards Compatibility

The transactions described in this SIP are an optional implementation, and are expected to be available to the user as a configuration option which is disabled by default.

This "notary participant" option, when enabled, will both generate the initial proposal transactions on the external blockchain and the subsequent signing process.

No participation is required and none will be taken without available funds in the litecoin service for the "bank" account name.

SIP 32: Multiple PoW Algorithm

```
Layer: Consensus (soft fork)
Author: Neo Natura < support@neo-natura.com >
Status: Proposed
Type: Standards Track
Created: 2019-04-20
```

Abstract

The SIP introduces a method to allow for additional PoW algorithms to be performed on the ShionCoin blockchain.

Motivation

Currently, only scrypt based mining hardware may be used in order to commit blocks to the ShionCoin blockchain. By allowing for multiple PoW algorithms to be used on the same blockchain, the burden of having to purchase specific mining equipment is lessened in favor of a more flexible software design. In addition, optional support for particular PoW algorithms on the ShionCoin colored alt-chains provides a means to further personalize their attributes. The algorithms that have been initially selected are based on either simplicity of design (for ASIC compatibility), SHA-3 competitiveness (for proven reliability), or broadness of usability (for popularity).

Specification

Additional PoW algorithm validation is provided by specifying a custom block version and by computing the difficulty in a predefined per algorithm ratio against the native scrypt difficulty. A 16-bit bitvector is added to the constant 0xE0000000UL in order to establish a block version to indicate which algorithm is being used. As such, consensus or versionbits cannot be specified when submitting a block with a PoW algorithm besides scrypt.

The initial algorithms to be provided on the mainnet blockchain, as part of a deployment consensus, are as follows:

- SHA256D
- KECCAK
- X11
- BLAKE2S
- QUBIT
- GROESTL
- SKEIN

Each colored alt-chain can be configured (upon it's genesis) to support, in addition to scrypt, the SHA256D, KECCAK, X11, and/or BLAKE2S PoW algorithms. Colored alt-chains do not require a consensus to use this functionality, but nodes cannot access an alt-chain which has options that are unsupported. Each algorithm is defined a hard-coded hash speed ratio in comparison to scrypt.

Only algorithms that have a faster hashing speed than scrypt are supported by this implementation. Note that the merkleroot for all PoW algorithms listed is generated using a traditional double-sha256 hash instead of the more complicated scrypt method. Segregated witness merkle root hashes are computed the same for all algorithms. All difficulties displayed through stratum or RPC are still shown in comparison to scrypt.

The block's "nBits" (compacted minimum difficulty) will reference the underlying PoW algorithm used. ShionCoin uses the Kimoto Gravity Well to determine the difficulty rate of newly generated blocks. When the difficulty of the next block is computed, the first block processed will be the last block in the chain that has been committed for the particular PoW algorithm desired to be submitted. As such, the time since the last block with the same PoW algorithm committed to the blockchain is used in order to determine the difficulty. Only the first block is filtered in this manner.

The remainder of the previous chain blocks used to compute the difficulty will be processed; regardless of their underlying associated PoW algorithm. The minimum difficulty is used if a block with that PoW algorithm has never been committed.

SIP 33: Dilithium Coin Address Signature

Layer: Consensus (soft fork)
Title: Dilithium Coin Address Signature
Author: Neo Natura < support@neo-natura.com >
Status: Proposed
Type: Standards Track
Created: 2019-08-30

Abstract

The SIP introduces a method to allow for a new pubkey coin address which can contain a dilithium-based signature.

Motivation

Currently, coin addresses in standardized virtual currencies rely upon the ECDSA signature method. This has been deemed insufficient in the eventual introduction of quantum computing. A quantum-safe signature method, dilithium, is proposed to be used as an configurable alternative.

The usage and methodology of the signing process is consistent with ECDSA. A pubkey is about 2.5k bytes and signature is also about 2.5k. The dilithium signing method is proposed to exclusively use the signature witness method having the affect of the longer signature not costing an additional tx fee.

Specification

The introduction of a new signing method implies the ability to handle multiple types of pubkey and signature data on the blockchain. The existing script operation OP_CHECKSIG is enhanced to verify against a dilithium signature when the dilithium witness version (14) is set.

The manner in which pubkey and script IDs is consistent with regular transactions. The witness v0 method is essentially identical to the dilithium witness program except that is verified using the dilithium-3 algorithm and that the public key and signature components of the segwit signature are larger in size. Note that the dilithium-3 signature, which is 2701, exceeds the traditional script maximum element size of 520 bytes.

The dilithium public key, stored in the wallet and used in transactions, is prefixed with a single byte set to the constants 0x3. This single-byte prefix is embedded for future extensions. Using the set of script ops above, the sender will include a keyid (28 bytes) and the receiver will include the full pubkey and signature.

The new signature method is exclusively isolated to a single witness program. The witness program version "14" is utilized in order to distinguish a dilithium signed transaction. Given a 20-byte key ID; a witness v14 destination address script would be "0x14 ". This is interpreted equivalent to a "OP_HASH160 OP_EQUALVERIFY" with the witness version being set to 14.

Note that a traditional witness script is not utilized (due to exclusive bech32 method). In addition, the signing method is reserved for use with the bech32 address format. As the bech32 specifies a particular witness program version, a receiver coin address can be specified that implies the signature will be stored on the segwit portion of the transaction (no counted against total block size) and that a dilithium algorithm must be used in order to verify the transaction as an subsequent input.

Deriving HD Keys

The implementation includes a HD (bip32) chain in order to derive dilithium based coin addresses. Each account is provided a master key

which uses a standard "hd key path" method in order to
provide derived keys for use with each user account.