PascalCoin Version 2

By Albert Molina (bpascalblockchain@gmail.com), Herman Schoenfeld (herman@sphere10.com)

preamble

PascalCoin is a next-generation cryptocurrency that pioneers a new tier of scalability comparable to the VISA network. PascalCoin paves the path towards "**Infinite Scalability**" and a new form of decentralised application coined "**Monetized API's"**. Version 2 of Pascal Coin addresses shortcomings in Version 1's account distribution model and delivers the key feature of **Checkpointing** (among other improvements). The addition of Checkpointing into PascalCoin now delivers the promise of **true** "**Deletable Blockchains**"**.** The blockchain in Pascal Coin V2 is now **permanently deletable from the Network itself** without affecting Proof-of-Work validation for new nodes. This means new nodes can join the network without the need to download the infinite history of blocks. Instead, they download the latest **Checkpoint** and a few dozen latest blocks in order to fully synchronize with the provably most-work-chain. This paves the path towards, what this paper coins, **"Infinite Scaling"**.

Introduction

PascalCoin is an innovative cryptocurrency that extends the blockchain-paradigm by introducing a new cryptographic structure known as the **SafeBox**. The SafeBox maintains a **ledger balance** rather than a ledger[2]. PascalCoin facilitates value-transfer between users by allowing them to transact funds (PASC) to and from accounts (PASA) much in the same way as traditional banking. Unlike most cryptocurrencies, PascalCoin accounts are simple and easy-to-remember (e.g. 1234-56) and not complicated strings which must be copy-pasted or scanned. Since only the ledger balance is required for consensus and not the full ledger, PascalCoin attains exponentially higher transaction throughput per unit of storage than UTXO-based cryptocurrencies. The design philosophy of PascalCoin is to take Bitcoin and rather than abstract and generalise as Ethereum does, simplify and refocus into a single-use-case-optimised currency.

The SafeBox is the ultimate source of truth in PascalCoin and provides a ledger balance of all users' funds. Structurally, it is like a spreadsheet where each row denotes a bank account (PASA) and each column denotes a property of that account (i.e. PASC balance, public key, etc). The "address" of an account is simply its index within the SafeBox (with an appended checksum). In addition, every 5 rows are grouped into an *Account Segment* sub-structure which corresponds to a block in the blockchain. Every time a new block is minted, the transactions/operations contained within that block are applied to the SafeBox resulting in a mutated state, and 5 new accounts are created. The resultant hash of the mutated SafeBox must then be referenced by the subsequent block in order to qualify as the next block.

Unlike traditional UTXO-based cryptocurrencies, the blockchain in PascalCoin is only used to mutate the SafeBox in a decentralised, ACID manner, not to serve as a source of funds. Whilst a Proof-of-Work blockchain is still required to facilitate Byzantine consensus (up to a checkpoint), it is not permanently required. As a result, the blockchain in PascalCoin is deletable.

The PascalCoin Value Proposition

Infinite Scaling

In the context of this document, Infinite Scaling is defined as "*the ability of a blockchain-driven network to achieve infinite running time on finite and constant storage".* This definition defines a theoretical limit which measures an upper-limit of a "***Throughput-Per-Unit-Of-Storage"*** KPI. Blockchain architectures with high Throughput-Per-Unit-Of-Storage result in high numbers of users with fast confirmation times and low-fees. Networks with low Throughput-Per-Unit-Of-Storage experience slow confirmation times, high-fees and admit an inherent ceiling of users.

How does PascalCoin achieve "Infinte Scaling"

Simply put, the blocks in Pascal Coin will be deletable past the checkpointing height of 100. Since new blocks will be appended to the top of the chain and old blocks deleted from the bottom, only a constant number of blocks will ever be required at any time. Checkpoints occur every 100'th block and are simply compressed SafeBox archives. When a new node joins the network, it only downloads the latest Checkpoint and a few dozen blocks. In addition, the SafeBox now contains block header information in every *Account Segment* sub-structure. This makes it possible for a node to independently compute and verify the cumulative work required to construct the SafeBox structure. It does this by:

- Checking that all block headers link transitively as a chain via the SafeBox
- Recalculating the Accumulated Work of the SafeBox using the proof-of-work payloads
- Verifying that the Accumulated Work of the SafeBox is the largest known in the network.

As a result, PascalCoin achieves exponentially higher Throughput-Per-Unit-Of-Storage than other cryptocurrencies, since node's only need to store the network throughput **not the aggregated network throughput**. In other words, PascalCoin stores the *flow of transactions* rather than the *history of transactions*. If the flow is constant, so is the storage. A caveat here is that the SafeBox does grow negligibly with every block, but always in constant amounts and irrespective of the quantity of transactions. For example, by the year 2072 the SafeBox will always be ~6 GB in size whether 1 transaction occurred or a googleplex.

Physical scaling limits

Since nodes only need to keep 100 blocks or so at any one time, PascalCoin allows for exponentially larger block sizes than current cryptocurrencies. For example, for the same amount of storage that a Bitcoin node consumes today, PascalCoin could theoretically sustain a blocksize of 5.4 GB with a throughput of **72,000 txn/sec.** Clearly there would be other bottlenecks such as signature verification and network overflow at that extreme scale, but it does highlight **the new tier of cryptocurrency** PascalCoin pioneers.

Why other coins cannot achieve similar scaling

Other cryptocurrencies cannot achieve this for two reasons. Firstly, they rely on the old block data to serve as the source of funds for new transactions in the form of UTXO's (Unspent Transaction

Outputs). Secondly, stake-proofs with in the Proof-of-Stake paradigm cannot be used retrospectively to aggregate the “total stake” staked to create that structure. In short, the SafeBox structure attains an intrinsic “difficulty” property proportional to the total work of the blocks used to create it, and cannot be easily forged. **This is only achievable in the Proof-of-Work paradigm**.

Whilst there are other approaches these cryptocurrencies could use such as pruning, warp-sync, "finality checkpoints", UTXO-snapshotting, etc, there is a fundamental difference. Their new nodes can only prove they are on most-work-chain using the infinite history whereas in PascalCoin, new nodes can prove they are on the most-work-chain **without** the infinite history. MimbleWimble is the closest proposal to achieve what Pascal Coin already does in terms of storage efficiency in UTXO-based cryptocurrencies.

Strong 0-Confirmation Guarantees

Since PascalCoin is not a UTXO-based currency but rather a State-based currency, the security guarantee of 0-confirmation transactions are much stronger than in UTXO-based currencies. For example, in Bitcoin if a merchant accepts a 0-confirmation transaction for a coffee, the buyer can simply roll that transaction back after receiving the coffee but before the transaction is confirmed in a block. The way the buyer does this is by re-spending those UTXOs to himself in a new transaction (with a higher fee) thus invalidating them for the merchant.

In PascalCoin, this is virtually impossible since the buyer's transaction to the merchant is simply a delta-operation to debit/credit a quantity from/to accounts respectively. The buyer is unable to erase or pre-empt this transaction from the network’s pending pool until it either enters a block or is discarded. If the buyer tries to double-spend the Coffee funds after receiving the Coffee but before they clear, **the double-spend transaction will not propagate the network** since nodes do not propagate a transaction if it double-spends a current pending transaction.

For even higher security guarantees, The PascalCoin Developers will soon roll-out a free and public “*double-spend-detection-service*” consisting of a JSON API that links to a several nodes geographically distributed throughout the world. For merchants who want high assurances for 0-confirmation payments, they can simply query this service after 10 seconds of receiving a PASC payment. If the service replies “No double spend detected” it means it is virtually impossible for a double-spend to occur since the majority of nodes are honest and will not propagate a double-spend attempt. The only way a double-spend can occur is if the buyer colludes with a malicious miner to mint his secret double-spend in the next block - a costly and unlikely proposition. As a result, the merchant attains **a very high assurance** that a 0-confirmation payment will clear as there is no double-spend anywhere in the world after 10 seconds and the majority of nodes are honest. **Good enough for coffee.**

No Need For Lightning Network

As a direct consequence of reliable 0-confirmation transactions, there is **no need for a Lightning Network** in PascalCoin since 0-confirmation transactions are faster and their security guarantees almost as good - sufficient for micro-payments and everyday-commerce.

Commoditization of Address Space

In almost all other cryptocurrencies, new users can simply create a new address for themselves at will. This creates an infinite address-space which can quickly bloat the blockchain even though the number of users remains constant. If the address space was instead made finite, it becomes a limited resource able to be commoditized. This is how PascalCoin accounts (PASA) operate. The accounts are limited, but any public key can be associated to it. This creates a natural space-saving mechanism since the chain is not littered with unneeded or used keys. It also disincentivizes spammers, since spammer accounts would be naturally limited and thus easily identifiable/blockable. Also, and most importantly, commoditization of the address space facilitates the SafeBox structure itself which is the key component to achieve "Infinite Scaling".

Account Names and Types

One of the key new features of PascalCoin is that accounts can have unique names which are publically visible, much in the same way as the domain names system. This allows a user to receive funds to their email address or chat moniker. It allows a shop to receive payments to their domain name or brand name. Payments still refer to accounts via numbers, but the name is used to lookup the account number just as a domain name is used to lookup an IP address.

More importantly, account names and types are serve a fundamental purpose in Layer-2 applications and Monetized APIs. For example, the account name could serve as a chat room name or a forum name. Account types further serve as a means to distinguish accounts for their use-case. For example, browsing accounts with type = 2 could be like browsing a list of chat rooms. How users interact with such Layer-2 applications is via Monetized API's described below.

Monetized API's

Due to reliable 0-confirmation transactions, PascalCoin allows a new form of decentralised application coined here as "**Monetized APIs**". In a Monetized API, PascalCoin accounts serve as *"ports"* that listen/send *"monetized-messages"* to other *"ports"*. It achieves this by repurposing an account as a named message-queue and by utilising Operation Payloads. In PascalCoin, operations can carry any arbitrary 256 byte payload of user-data. Payloads can be public or encrypted.

This unique capability allows operations to embed "**Layer 2 protocols**" in much the same way that HTTP lives inside TCP. The difference here is that the protocol messages carry a **financial weight**, and as a result, can be used to conduct granular economic communication suitable for algorithmic/autonomous/micro-commerce scenarios. For example:

- **Pascal Chat**: An account can serve as a chat room where the the account name is the room name. Operations payloads to the account can contains the user's chat message. The users handle would simply be the sender's account name. These chat rooms can be used for trading goods and services in a decentralised manner. Anonymity can be enhanced via other Layer-2 applications. Users can settle payments for goods by attaching funds to private messages between themselves.

- **Monetized Content:** A PascalCoin browser-plugin that pays content providers for content on-the-fly. The payment itself contains info to allow the server to unlock the content for the browser. This could be used for monetizing news, blogs, ebooks and social media content.

- **One-click e-commerce:** a one-transaction e-commerce site who's shopping cart is reduced to a single "*Payload Code*". The buyer merely sends a PASC payment containing the Payload Code, and when received by the merchant the order is executed for the buyer.. No credit card numbers or complex payment-gateway integrations were required.

- **Anonymity Mixers**: accounts can receive funds from other accounts and then then re-send those funds using complex financial routing information encrypted within the sender's Payloads. The mixer can split/delay/relay/loop payments arbitrarily thus sufficiently obfuscating the sender, receiver and quantity from taint analysis.

- **Layer-2 Side-Chains**: since messages to/from an arbitrary account X can be used as a message queue, it is possible to construct Layer 2 Protocols for managing a side-chain by monitoring messages to X. The side-chain's validity/state is governed by the Layer-2 protocol embedded within these Layer-1 payloads. The owner of X serve as an authority on a mono-federated side-chain suitable for Dapps. Owner-free (or non-federated) side-chains can be constructed by associating a proof-of-burn key on an account. **Federated side-chains** will be possible when Schnorr multisig is implemented whereby members of the federation will be comprised of the Account signatories.

An example of an actual Monetized API, and the first one ever created, is GetPasa.com. It works by listening on account 77-44 for incoming transactions of 10 PASC or more. When a transaction is sent to 77-44 containing 10 PASC or more, it's payload is examined for the presence of a public key. If nothing is found, transaction is discarded. If a public key is found, the service then finds a free account in it's inventory and send it to the key it found. This allows new users to purchase an account in a one-step process that sends a single message containing the order and payment combined.

Assets, Sub-Tokens and Smart Contracts

By leveraging PascalCoin's Layer-2 architecture, it is possible to achieve Assets, Sub-Tokens and Smart Contracts in the same way Rootstock achieves Ethereum over Bitcoin. Running Ethereum Virtual Machine over PascalCoin would be possible by maintaining a side-chain pinned to an account (as Rootstock does). Transactions to this account would embed Layer-2 protocol commands that govern the EVM side-chain. Interestingly, Sharding could be achieved easily since independent EVM side-chains could run bound to separate accounts. Inter-shard communication would simply be transactions between these accounts. The rest of the network would not really be impacted by the large volume of transactions since the natural process of Checkpointing discards these transactions after 100 blocks. Only side-chain users would bother to record all account transactions in order to validate their side-chain.

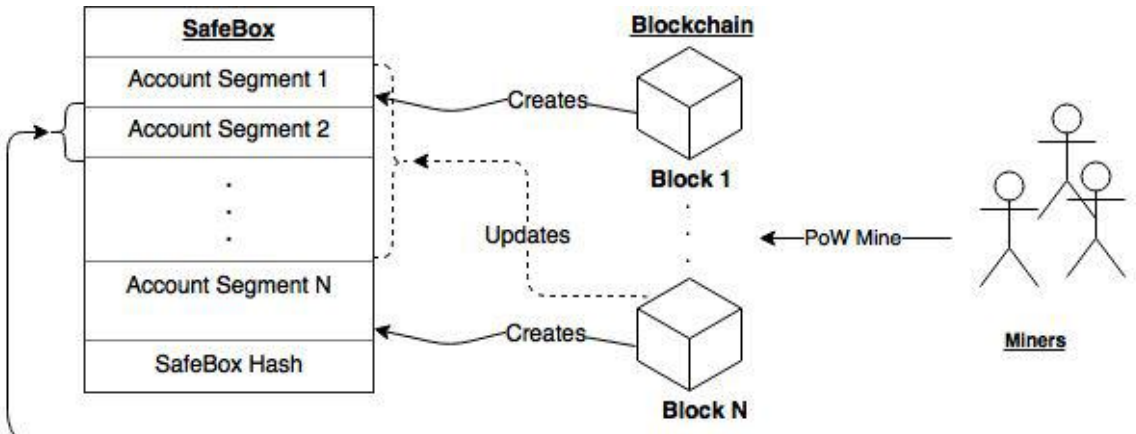
Self-Funded Community & Written in Free Pascal

PascalCoin was a 100% fair launch without any premine, ICO or investment rounds. The PascalCoin developer community are independently wealthy, self-funded evangelists who develop and promote this game-changing technology. The Pascal programming language has evolved far beyond the days of Turbo Pascal. Free Pascal is a modern object-oriented language with advanced features such as generics. It was originally designed as an alternative to C, and with it's modern advances and upgrades, has become a great language for writing high-performance, cross-platform native code.

PascalCoin Architecture

The SafeBox

As opposed to a series of blocks containing transactions records going from one address to another, PascalCoin uses 2 components: the SafeBox (containing all current account balances) and blocks linked together to form a blockchain. Just as with Bitcoin, mining nodes are responsible for creating a new block. All nodes update their copy of the SafeBox independently of each other when new blocks are announced. As part of this task, nodes need to update the balances (and other fields) of existing accounts based on operations in the block as well as create a new Account Segment containing brand new PascalCoin accounts awarded to the winning miner.



SafeBox Hash = SHA256(Segment1.Hash + Segment2.Hash + ... + SegmentN.Hash)

Account Segment 2

Acc. Number	Balance	Public Key	Name	Type	N Op	Timestamp	Hash
5	*1231.1214*	*PubKey*	user23@email.com	0 - User Account	4	Unix Time	H2000
6	*9.0000*	*PubKey*	@SlackStyle	1 - PascalChat Room	2	Unix Time	H2001
7	*999.0000*	*PubKey*	#SlackStyle	0 - User Account	1	Unix Time	H2002
8	*1231.1231*	*PubKey*	domainstyle.com	2 - PascalShop	3	Unix Time	H2003
9	*5555.1111*	*PubKey*	<empty>	0 - User Account	1	Unix Time	H2004

Segment Hash = SHA256(AccountNumber ++ Balance ++ *rest of fields*)

First Block

Before the first block is created, the very first Safe Box number 0 (genesis SafeBox) is created which does not have any account in it. Miners will seek a new block for the block chain via the proof of work where the hash of genesis SafeBox is used as input. Once this first block is created, a corresponding new version of the SafeBox is created along with a Account Segment containing N new accounts, where N is defined by the protocol (N is set to 5 in the current PascalCoin Version 2). Now the miners will start working on the next block of the blockchain in order to generate the next version of the SafeBox which will include new pending operations circulated by nodes.

Operations

Similarly to Bitcoin and other cryptocurrencies, Blocks in PascalCoin are containers for transactions called “Operations”. They are referred to as Operations as they are generalised transactions and perform more than simply transfer funds between accounts. For example, there are operations to change an account's key, to change it’s name or to mark it for sale, etc. The most important and common operation is the Transaction operation which which transfers funds between accounts.

Accounts

The SafeBox is essentially a list of accounts. Accounts contain funds, the public key of the owner, a unique name and a type field.

Element Name	Data Type	Description
Account number	Unsigned 32 bits	Ordinal number identifying the account, this is never modified.
Public EC Key	Public Key (type, X and Y) (between 66 and 200 bytes)	The public key is the PIN of the account. This value can be changed at any time but only by the owner of the corresponding private key of the currently assigned Public Key.
Balance	Unsigned 64 bits	Current account balance
Updated block	Unsigned 32 bits	Number of the last block of the block chain that modified this account. This value helps identify stale unused account.
N Operation	Unsigned 32 bits	Incremental value indicating how many transactions were made with this account and ensures that order of operations are unique and therefore not duplicated.
Name	RawBytes	A unique and public name for this account. By default it is blank. The name is encoded in PascalCoin64 encoding.
Type	Word	Used to differentiate accounts for different

		purposes. As Layer-2 protocols are established, this value will become useful. Example, Type=2 may be reserved for 'Chat Channels' and Type=3 reserved for 'Online Shop', etc.

Account Segments

The accounts in the SafeBox are grouped in segments to form what is called an Account Segment. Account Segments are generated every time a miner appends to the SafeBox through mining. In other words, the SafeBox is extended atomically with a new Account Segment each time a new block is being mined.
The content of each Account Segment are as follow:

Element Name	**Data Type**	**Description**
Block number	Unsigned 32 bits	This is equivalent to the block number in the block chain (see later section).
Accounts	Array of N accounts	Fixed Array (size N) with account numbers that are generated by this block. N is set to 5 in the current PascalCoin Protocol version but may be increased and/or made dynamic in future versions.
Timestamp	Unsigned 32 bits	Unix Timestamp when generated. This value is unchanged forever.
Account Segment Hash	32 bytes	Hash value of this block. It changes every time any of the accounts in this Account Segment changes (either balance adjustment, or Public EC key changed). This validates and secures the integrity of this block.
Block Header	**~180 bytes**	**This is new in V2.** This data allows a node to recompute the total work used to construct the SafeBox without the need of the blocks.

In addition, the SafeBox contains a checksum created as an aggregate hash of all Account Segment hashes. This value is known as the SafeBox Hash and is appended immediately after the last Account Segment in the SafeBox. The next block must also reference this SafeBox Hash in order to be valid.

The Blockchain

Just as with Bitcoin, the integrity of the financial data is secured by Proof Of Work using a series of blocks chained together. Also like Bitcoin, these blocks contain a list of transactions used to mutate the financial state. However, unlike Bitcoin, a block does not directly reference the previous block.

Instead it references the previous SafeBox Hash, which transitively links to the previous block via the Account Segment for that SafeBox Hash. As described before, when a miner wins the Proof of Work, it will publish its block resulting in an updated SafeBox that will contain a new Account Segment with N new accounts in it. The miner is awarded these new account by being assigned to the miner's public key.

The Block structure contains the following fields:

Element Name	Data Type	Description
Block Number	Unsigned 32 bits	Block number generated by the miner.
Account key	Public Key (type, X and Y) (Between 66 and 200 bytes)	The miner list its public key which will be assigned to the N new accounts created in the Block Account.
Reward	Unsigned 64 bits	The miners reward. Set to 100 Pascal initially and halving about every 4 years.
Fee	Unsigned 64 bits	Some of all transaction fees the miner is collecting by mining this block of the block chain and its corresponding Account Segment.
Protocol version	Unsigned 16 bits	Protocol version
Protocol available	Unsigned 16 bits	Protocol number that can apply the miner owner of this block (for future compatibility purpose on protocol upgrades)
Timestamp	Unsigned 32 bits	Unix Timestamp this block was created
Compact target	Unsigned 32 bits	This is the difficulty level the miners must obtain in the proof of work.
Nonce	Unsigned 32 bits	The nonce used by miners to get the required result with the Proof of Work. (ie: with the hash having a number of leading 0 bits just as with Bitcoin)
Previous SafeBox Hash	32 bytes	Value of the previous SafeBox hash from which the next SafeBox version is created from.
Operations Hash	32 bytes	Merkle Tree Hash (see below)
Proof of Work	32 bytes	The hash of this block, used for the Proof Of Work.

Protocol Version 1 Limitations

PASA Distribution Model

In PascalCoin, every time a block updates the SafeBox new accounts are appended at the end of the SafeBox. These accounts are awarded to the miner as a "secondary reward" alongside the usual block reward. These newly generated accounts are then distributed to ordinary users via market mechanisms. Actual ownership exchange of an account occurs by updating the key associated to that account, via a blockchain operation. The bearer of the private key which corresponds to an accounts public key is the authorized owner of that account. Since the introduction of PascalCoin, this distribution workflow has proven an impediment to adoption due to:

- **Exchange-PASA Problem:** Most users acquire their PASC from exchanges, not through mining. Typically, after a user buys PASC on an exchange they then attempt to withdraw from that exchange into their wallet only to find out they need a PASA. Since PASA assets are not traded on established exchanges, users resort to OTC markets in chat rooms/forums prone to human error and fraud. Exchanges are reticent to trade PASA due to their indivisible and non-fungible properties. PASA assets mirror numismatic gold markets more than bullion. As a result, PASA are better suited for auction-style markets (such as PascWallet.com), placing them beyond the reach of exchange users at this point in time. Protocol V2 addresses this by facilitating provably safe PASA exchanging directly within the protocol. First time users can also easily acquire an account via Monetized API's like GetPasa.com.
- **Miner Hoarding:** Due to the competitiveness of mining, most mining is now pooled collectively into mining pools. Typically, individual pool workers are awarded their yield in proportion to their hashing power contributions. In the case of PASA, this has proven burdensome for pool operators since PASA are indivisible and non-fungible assets. The additional effort required by pool operators to fairly distribute PASA has actually prevented distribution entirely and resulted in significant hoarding. Protocol V2 now provides an effective in-protocol means to distribute PASA.

PascalCoin Protocol v2 addresses this shortcoming by implementing decentralised account sale, settlement and exchange operations within the protocol-itself. In combination with PASA exchanges such as PascWallet.com and GetPasa.com, the account distribution problems of Version 1 are resolved.

New Nodes Required Infinite History

Even though Version 1 allowed nodes to delete blocks after they've processed them, new nodes had to download from block 1 in order to prove they were on the most-work-chain. As a result, the major promise of a Deletable Blockchain was merely the equivalent of pruning. Version 2 solves this issue by now making a SafeBox a self-verifiable in terms of Proof-of-Work.

Protocol Version 2 Changes

Improved Difficulty-Calculation

Due to the volatile nature of cryptocurrency valuations, miners will tend to migrate between coins en masse based on the yield-rates of coins. As miners change coins, time is required for the difficulty to adjust. Cloud-mining has exacerbated this issue requiring consideration in difficulty calculation. PascalCoin V2 has enhanced the difficulty calculation algorithm to to better adjust with near-term sinusoidal volatility resulting from rapid increases and/or decreases in hashpower. As a result, the blockchain will be more stable, block time and daily supply will be more accurate.

Account Names & Types

Accounts have been updated to include names and type. An account name is case insensitive and must be 64 or less characters in length. An account name must be unique in that no other account has registered that name, or there are no pending operation which register that name. No block can contain a name registration for the same name, or for an already registered name. In order to avoid spoofing of account numbers, account names must never start with a digit. The character set permitted for account names is called “PascalCoin64”.

Accounts also now contain a 2-byte Type field used to distinguish accounts for various Layer-2 applications and monetized APIs. The range of values for account type are between 0 and 65535. Account type 0 is reserved as the default type for standard “User Accounts”.

New Operation: Change Account Info

This new operation allows a user to change an account name and type. The account name is encoded in PascalCoin64 and must be unique or empty. If a Change Account Info operation registers a name that’s already registered, the operation is invalid. If there is another pending operation for the same name, the operation is discarded and not propagated by nodes. If a block contains two valid operations registering the same name, the block is invalid. Empty account names are exempt and any account is permitted to be nameless. This operation contains the following properties:

- Account number - the account who’s name will be changed
- Account Name - the PascalCoin64 encoded name for the the account.
- Account Type - an WORD value depicting the type of account
- Network Fee - the fee paid for this operation
- Fee Paying Account - the account which will be paying the fee, this account must have the same public key as the account being changed
- Signature

PascalCoin64 Encoding

The encoding allows case-insensitive, whitespace-free names names capable of supporting emails, domain names, slack channels/users such as *herman@sphere10.com, #PascalCoinDevelopment , CoffeeTables.com, @MySlackStyleName, AnythingWithoutSpaces123!!%6.* The character set is defined as:

abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()-+{}[]_:"|<>,.?/~

Checkpointing

Version 2 introduces the concept of Checkpointing. Since all (honest) nodes will always have the same SafeBox after every block, it is possible to snapshot the SafeBox at certain intervals. These snapshots are called "Checkpoints" and they are created every 100 blocks. The key innovation in V2 is that the SafeBox now contains the block header information in every *Account Segment* sub-structure. This makes it possible for a node to independently compute and verify the cumulative work required to construct the SafeBox structure. It does this by:

- Verifying that all block headers link transitively as a chain via the SafeBox
- Recalculating the Total Work used to build the SafeBox using the proof-of-work payloads now present in the Account Segment sub-structures.
- Verifying that the Total Work of the SafeBox is the largest known in the network.

The consequence of this is that Checkpoints can now be distributed safely between nodes **without the need for blocks to verify them**. In coming releases, a vastly improved checkpoint distribution infrastructure will be soft-forked in called **Checkpoint Torrenting**. The full process will work as follows.

The Checkpoint is persisted as multiple same-sized chunks of raw data. When a new node comes up, it will query from multiple nodes for the latest Checkpoint information consisting of

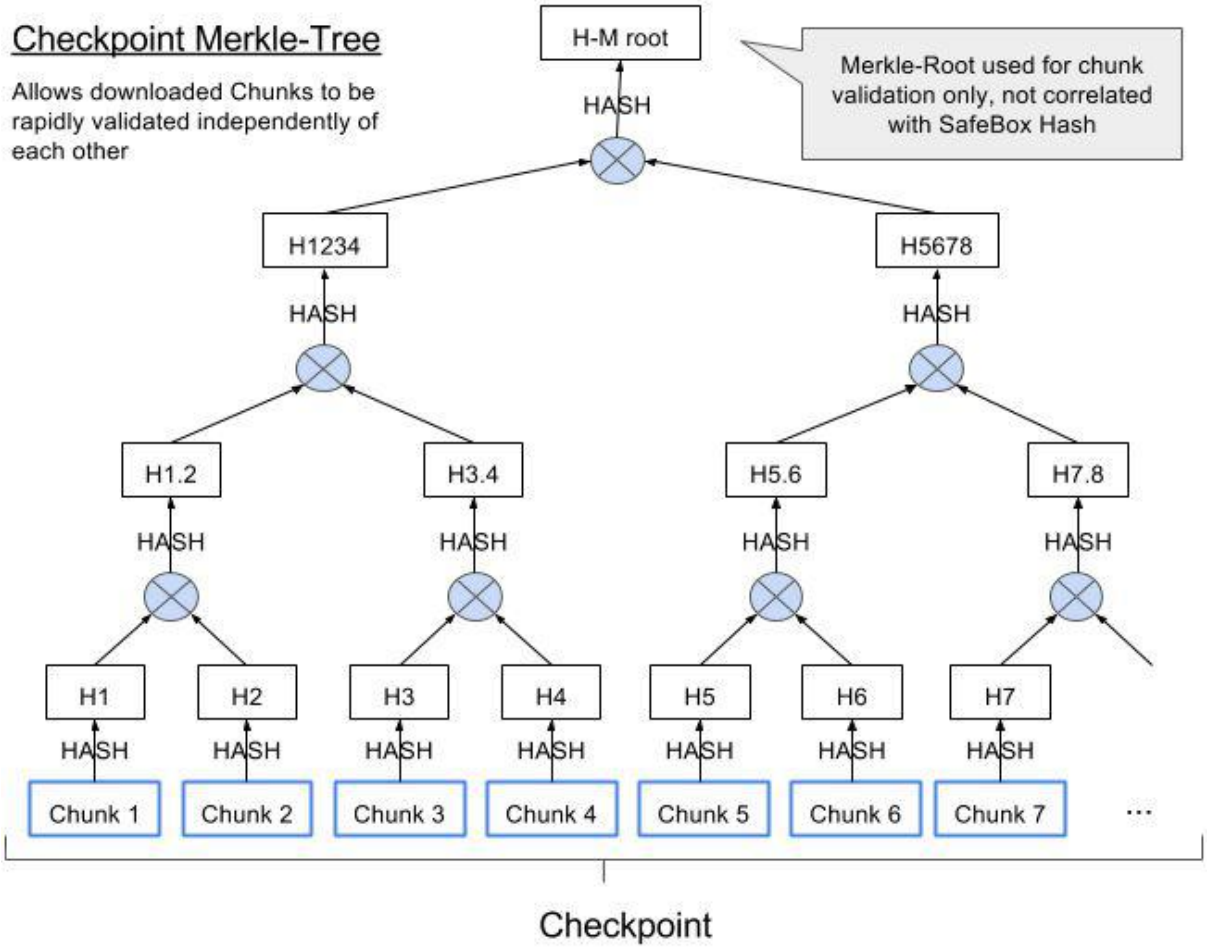
- The checkpoint block number
- SafeBox Hash of the Checkpoint
- Total work of the Checkpoint
- Merkle-tree of the Checkpoint chunks

The new node can then determine the correct Checkpoint to download by simply selecting the one advertised with the most total work. The node then downloads the Checkpoint chunks simultaneously from many nodes.

During transmission, chunks are significantly compressed which greatly improves synchronization speed and reduces network traffic. Chunks are then independently validated using the provided merkle-tree (see diagram below). Once all chunks are downloaded, the Checkpoint is reconstructed and verified for structural integrity. The total work is then recalculated and compared to the advertised total work from the beginning of the synchronization process.

If the Checkpoint turns out to be invalid for any reason, it is discarded and the advertising nodes blacklisted. The synchronization process then repeats ad infinitum until a valid checkpoint is

downloaded. If the Checkpoint is valid, it is adopted as the SafeBox and the usual block synchronization process resumes. At most, a new node will only ever need to download 100 blocks in order to be fully synchronized.



In-Protocol PASA Exchange

Protocol v2 provides the following new operation types to facilitate the secure sale, purchase, settlement and exchange of PASA assets between users:

- List Account For Sale
- Delist Account
- Buy Account
- Change Info

and amends the following operations

- Transaction *(this is the standard operation used to transfer PASC)*

New Operation: List Account For Sale

The purpose of this new operation is to allow an owner of an account to designate an account "For Sale" in much the same way an owner of a house may place a "FOR SALE" sign outside their house. Accounts can be marked for **public sale** or for **private sale**.

For a **public sale**, anyone can purchase the account by executing a corresponding *Buy Account* operation containing the correct funding. The first user to execute a valid *Buy Account* operation will become the owner of that account. If multiple *Buy Account* operations arrive simultaneously, the miner selects one and discards the rest. Any excess funds remaining after a *Buy Account* operation are credited to the purchased account. If a *Buy Account* operation fails for any reason, those funds are never debited from the purchases account.

For a **private sale**, the seller designates the buyer's public key within the listing and includes a timeout period to allow the buyer to settle the purchase. The buyer may settle the purchase by executing a single *Buy Account* operation, within the timeout period, with the correct funds. Optionally, the buyer may send one (or many) standard *Transactions* into the account until the funding is met, at which point the purchase is settled.

The properties for this operation type include:

- The account number to be sold
- The sale price
- The sellers account number receiving payment
- Flag indicating whether private or public sale
- Timeout period (*private sale only)*
    - This value is a block number which the purchase must be completed before or on.
    - During this period, the account is frozen and only able to receive funds, in order to protect the buyer.
    - After this period, the account owner can delist this account and consume funds.
- Buyers public key *(private sale only)*
- Seller's signature

New Operation: Delist Account

This new operation allows an account previously listed for sale to be delisted. If an account is listed for public sale and is settled prior to the execution of a *Delist Account* operation, then the *Delist Account* operation is discarded. If an account is marked for private sale then it cannot be delisted during the timeout period, only after the timeout period. If an account's timeout period has elapsed and purchaser has not settled the purchase, the account will continue to be listed for sale until either it is purchased or delisted.

The properties of this operation type include:

- The account number to be delisted
- Seller's signature

New Operation: Buy Account

This new operation allows a user to purchase and settle an account listed for public or private sale. For a public sale, the operation must include the funding equal to or greater than the listing sale price. If the funding is less than the listing price, the operation is considered invalid and discarded. If the funding is greater than the listing price, the excess funding will be credited to the purchased account after settlement.

This operation contains the following properties:

- Buyer's account number that will fund the purchase
- Account number being purchased
- Funding
    - Must be equal to or greater than the selling price, not less.
    - Funds already present in the account are deducted from the sale price.
    - Excess funds are credited to purchased account
- New public key for account
    - For a private sale, this must match the public key in the listing, otherwise will be discarded
- Buyer's signature

Modified Operation: Transaction

Users can optionally settle a private account sale by issuing standard transactions into the account being purchased. The existing *Transaction* operation type is modified as follows. If a transaction is crediting an account listed for private sale and the resulting balance is greater than or equal to the sale price, the account is purchased as if a *Buy Account* operation were executed. The account's resulting public key will change as per the buyer's key specified in the listing **irrespective** of the origin of the transaction. Also, an amount equalling the sale price specified in the listing is transferred to the seller's account specified in the listing. This update allows new users with no account but with coins in an exchange to withdraw their funds into an an account they've negotiated for.

Accounting Rules

The following accounting rules govern how all the interacting account's balances are mutated pre and post account purchase/settlement.

Let A = account balance pre-settlement (of account being purchased)
Let B = buyer's account balance pre-settlement (the account funding the purchase)
Let C = seller's account balance pre-settlement (the account receiving the funds)
Let T = funding amount (the PASC used to purchase account)
Let P = listed sale price of account

- Funding required to settle purchase = P - A
- Account balance post-settlement = A + T - P
- Buyer's account balance post-settlement = B - T
- Seller's account balance post-settlement = C + P

User Workflows

This section illustrates common use-cases addressed by Protocol V2 updates.

New PascalCoin User

Bob is a new PascalCoin user and wishes to hold his coins in his own wallet.

- Bob buys PASC on an exchange
- Bob installs the PascalCoin wallet and creates his key (private/public pair)
- Bob finds a Monetized API provider to provide an account
- Bob withdraws from exchange to Monetized API using his public key, and he receives account with funds.

User Wants Special Account

- Bob doesn't like his account number and wants a new account with his birthday, 1988.
- Bob finds Alice is selling account 1988 on an exchange and agrees to buy from her.
- Alice executes a *List Account For Sale* operation specifying Bob's public key, the agreed price and a settlement/timeout period.
- Bob's wallet recognizes Alice's private listing and prompts Bob to settle within the timeout period.
- Bob withdraws funds from the exchange directly into the listed account
    - This transaction does not require a payload
- Once the account receives Bob's funds the public key is changed to Bob's key and the funds transferred to Alice's account.

Miner Selling Accounts

Bob is a PascalCoin miner with many accounts he intends to sell.

- Bob executes a bulk *List Account For Sale* operation containing all said accounts
    - He sets a common price, applied to each account
- All selected accounts are now listed for public sale
- Over time, users purchase these accounts without further interaction from Bob
- Bob then decides to keep unsold accounts and executes a bulk "Delist Account" operation

Account Speculator Selling Rare Accounts

Moby is a PascalCoin account collector who has acquired many rare account (e.g. 1111 and 2222) and intends to sell them individually at high-prices.

- Moby lists his unique accounts on a 3rd party PASA auction exchange
    - **Bob does not transfer account ownership to the exchange**
- Alice bids for and wins Moby's account action for 1111
    - **Alice does not deposit funds on the exchange**
- Moby executes a *"List Account For Sale"* operation for account 1111 specifying Alice's public key, a 1 day settlement period and the agreed price of 100 PASC
- Alice's wallet detects account 1111 listed for sale exclusively to her for 100 PASC

- Alice executes a *"Buy Account"* for 1111 containing 110 PASC
- Alice becomes the owner of 1111 which now contains 10 PASC

Road Map

[✓] In-Protocol PASA Exchange
[✓] Deletable Blockchain
[✓] Checkpointing
[ ] PascalCoin Improvement Proposal process implementation (PIP)
[ ] OS X Wallet
[ ] iOS & Android Wallet (Native)
[ ] Checkpoint Torrenting
[ ] Ultra-high Throughput Fine-tuning & Optimization
[ ] V3-alpha testnet (100+ txn/sec on fixed storage)
[ ] Multisig (N-N Schnorr aggregate signatures)
[ ] Full C# Implementation (Community driven)
[ ] Monetized API Infrastructure For Service Providers (Community driven)
[ ] Pascal Chat (Community driven)
[ ] Double-Spend-Detection-Service (for highly-reliable 0-confirmation transactions)

Acknowledgements

The authors appreciate the suggestions and editing of this whitepaper from Phil Champagne (BookOfSatoshi@gmail.com) author of Book Of Satoshi.

References

[1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System"
https://bitcoin.org/bitcoin.pdf

[2] Albert Molina, "Pascal Coin: Crypto currency without need of historical operations"
https://github.com/PascalCoin/PascalCoin/raw/master/PascalCoin%20White%20Paper%20-%20EN.pdf

