Updated: January 2019, Version: 4.0

PascalCoin

Albert Molina bpascalblockchain@gmail.com

Herman Schoenfeld herman@sphere10.com

Whitepaper authored by Tyler Swob

PascalCoin is a next-generation cryptocurrency that pioneers a new tier of scalability comparable to the VISA network. Enabled by the SafeBox technology to become the world's first blockchain independent of historical operations, PascalCoin possesses unlimited potential. PascalCoin paves the path towards "Infinite Scaling" by allowing new nodes to 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."

TABLE OF CONTENTS

INTRODUCTION

PascalCoin is an innovative cryptocurrency that extends the blockchain-paradigm by introducing a new cryptographic structure known as the SafeBox [1]. The SafeBox is the ultimate source of truth in PascalCoin and maintains a ledger balance of all users' funds rather than the full ledger. 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. Pascal balance, public key, etc.). The "address" of an account is simply its index within the SafeBox (with an appended checksum). Every time a new block is minted, the transactions or operations contained within that block are applied to the SafeBox resulting in a mutated state. 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. 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 capable of being deleted without any security compromise.

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 [2] and rather than abstract and generalise as Ethereum does, simplify and refocus into a single-use-case-optimised currency with advanced features that include but are not limited to the following:

- Instant, zero-fee transactions
- Theoretical limit of 72,000 TPS
- Infinitely scalable smart contracts
- Comprehensive anonymity
- Addresses are simple account numbers (e.g. 1000-84) that can be associated with a name
- Infinitesimal storage requirement for nodes
- ASIC & GPU resistant algorithm

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 "INFINITE SCALING"?

Simply put, the blocks in PascalCoin will be able to be deleted 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 100th block and are simply compressed SafeBox archives. When a new full node joins the network, it only downloads the latest checkpoint and a few dozen blocks. In addition, the SafeBox 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 a node only needs to store the network throughput and not the aggregated network throughput. 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.

WHY OTHER COINS CANNOT ACHIEVE SIMILAR STORAGE EFFICIENCY

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 full 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 PascalCoin already does in terms of storage efficiency in UTXO-based cryptocurrencies.

EMPIRICAL TRANSACTIONS PER SECOND

In a typical blockchain implementation like Bitcoin, there are three bottlenecks which determine the processing speed:

- Network throughput – number of how many transactions can flood-fill network per second
- Verification throughput – number of signature and consensus checks per second
- Blockchain throughput – number of transactions append to the blockchain per second

The network throughput of PascalCoin and a traditional blockchain is similar because the net propagation speed is calculated based on the amount of bytes needed to communicate operations between nodes. The verification throughput of PascalCoin and a traditional blockchain are also similar. However, the blockchain throughput for PascalCoin is on an entirely different level thanks to PascalCoin's larger block size and its differing mechanism for double spend protection.

PascalCoin allows for exponentially larger block sizes than current cryptocurrencies since nodes only need to keep 100 blocks or so plus the SafeBox checkpoint. This in turn increases the blockchain throughput. More details on the larger block size are given in the next section about the theoretical limit of transactions per second.

The double spend protection mechanism entails searching a given blockchain's database (usually based on UTXO) for whether the previous transaction has been spent or not. With the SafeBox, this process is as quick as simply viewing the signer's account state. Thanks to how the SafeBox in its entirety is stored compactly in memory, this process is almost instantaneous in PascalCoin. In a traditional blockchain, this process is slower because it needs to search in a database saved on disk. The difference between the time needed to search in a saved database on disk (UTXO) versus searching in memory (SafeBox) is significant, leading to a nearly instant transaction speed in PascalCoin.

Multithreading is used to execute processes in parallel using logical CPUs. A single low-ended CPU will process at least +300 operations per second. If 4 CPUs are available, the speed would be at least 4*300 = 1,200 operations per second and would be much higher on modern CPUs. Most tests with a 4-core CPU indicate the maximum speed reaching about **1,600 transactions per second**. This reflects live performance on the main net, given that the proportion of slow nodes is relatively small compared to fast nodes. 1,600 transactions per second as an empirical approximation is among the highest figures attained in any actual blockchain tests on a main net to date – if not the highest of all.

THEORETICAL TRANSACTIONS PER SECOND

The theoretical limit for transactions per second is much higher than 1,600 transactions per second and is primarily based on PascalCoin's block size. For about the same amount of storage that a Bitcoin node consumes today, a PascalCoin could theoretically sustain a block size of 5.62 GB (compared to Bitcoin's 1 MB block size) with a maximum blockchain throughput of **72,000 transactions per second**. The maximum blockchain throughput is calculated as follows:

$$\text{Maximum blockchain throughput} = \frac{\text{Maximum block size}}{\text{Average operation size}} * \frac{1}{300}$$

The maximum block size of 5.62 GB is assumed and the average operation size is 262 bytes. The expression is then divided by 300, the number of seconds in a 5-minute block. This yields the 72,000 transactions per second as a theoretical limit.

The theoretical upper limit for PascalCoin's total throughput is the minimum of the network throughput, verification throughput, and blockchain throughput. This minimum amounts to 72,000 transactions per second assuming the network and verification throughputs are infinite. The theoretical rationale for assuming network and verification throughputs as infinite is how the blockchain throughput always has a hard, albeit changeable, limit whereas other throughputs will improve constantly over time.

It should be emphasized that PascalCoin's blockchain throughput is technically unbounded. The maximum number of operations per block is not fixed and could be increased to accommodate any level of mass adoption without any undue burden on nodes because they only need the last 100 blocks and the latest SafeBox checkpoint. Such an increase in the maximum number of operations per block would lead to a higher block size and consequently a higher blockchain throughput. The maximum block size of 5.62 GB from above was arbitrarily chosen as a reasonable example of PascalCoin's storage capacity but if the empirical transactions per second, 1,600, remains fixed then the maximum block size would be about 0.13 GB. Concretely, storage space for the last 100 blocks is a soft constraint on PascalCoin's otherwise infinite blockchain throughput.

PascalCoin's blockchain throughput from its efficient memory usage works in tandem with the nearly instant transaction speed of the SafeBox model to enable infinite scalability. It is estimated that PascalCoin will achieve 100,000 transactions per second as a theoretical limit (assuming a block size of 5.6 GB) with extensive core factoring, optimizations, and hardware configurations. PascalCoin's total throughput is essentially only limited by Moore's Law, storage space, and a highly decentralized architecture.

All the specifications above are for layer -1. With a layer-2 implementation, PascalCoin's scalability will be enhanced even further. More information on PascalCoin's layer-2 specifications will be released shortly.

CRYPTOGRAPHIC INTEGRITY AND SECURITY

The SafeBox preserves the cryptographic integrity of the full blockchain even though nodes are not required to store the full blockchain. That follows from the fact that the SafeBox contains all the block headers used to construct that SafeBox within the SafeBox itself. Each block header makes a hash commitment to the previous SafeBox state (i.e. the state of all accounts at that point in time) and also the previous block header. In this manner, the state and its evolution are preserved by using the difficulties in the block headers; the total work used to evolve that state can then be calculated. Nodes will only adopt the SafeBox with the most work.

A distinction should be made between cryptographic integrity, the proof that the SafeBox is hashed correctly from block 0, and cryptographic security, the number of blocks that need to be re-mined in an attack. The SafeBox retains the full cryptographic integrity and retains 99% of the cryptographic security of a full blockchain. The latter is due to the state-attack vector as explained in the following.

With regards to the state-attack, a distinction should be made between a double-spend and a state-attack. A double-spend refers to an attack involving majority hashpower and rolling back already accepted transactions but not hacking account balances as stored in the SafeBox. A state-attack refers to corrupting a balance and trying to cover it up by re-mining blocks in a way that other nodes cannot detect the hacked balance. With a double-spend, PascalCoin's security parallels to that of a secure Proof-of-Work UTXO blockchain and is a function of hashpower. The state-attack, however, is unique to PascalCoin and requires an attacker to re-mine the network median history to succeed. But compared to the standard double-spend attack with hashpower majority, the state-attack is extremely difficult and unrealistic to execute.

If a state-attacker naively alters an account balance then it becomes corrupt since the tip block has an invalid SafeBox hash. If the attacker tries to re-mine that tip block faster than the rest of the honest network such that it links to its corrupted SafeBox, then the honest network can detect the tip block is invalid since they have the last 100 blocks.

If a more sophisticated state-attacker attempted to re-mine the last N blocks, and do so in a manner that it linked to his corrupted SafeBox state, then it can only fool other nodes if and only if such nodes kept less history (N-1 blocks) and the falsified SafeBox had more aggregated work than the honest SafeBox. However, this is not enough to fool the entire network since there exists archival nodes which keep the entire history, and many nodes keep large numbers of blocks. So the theoretical minimum number of blocks an attacker needs to re-mine is N=100 but in practice, it is far higher. If the network keeps a median history of 1000 blocks, then the security of the SafeBox is equivalent to a blockchain of 1000 blocks.

Formally, the SafeBox model is as secure as the median block history of the entire network with regards to the state-attack but additional security parameters as follow substantially increase the SafeBox's security:

- The amount of difficulty required to re-mine a history grows exponentially. This means it's exponentially harder to re-mine the last 3 blocks than it is the last 2, and so on.
- PascalCoin uses a 5-minute block time. Re-mining 100 blocks with a 5-minute period is far more difficult than, for example, 100 blocks with a 30-second period.
- Random Hash, an innovative ASIC & GPU resistant algorithm, serves as a powerful deterrent on hashpower monopoly.
- Since nodes require the last 100 blocks minimum, the "network median history" will never fall below 100. In practice, due to the presence of archival nodes which maintain full-history and long-running nodes which maintain large histories, the "network median history" will always be significantly larger than 100.
- Current and upcoming synchronization implementations for nodes mean that a state-attacker would need to re-mine far more than 100 blocks. For example, the default setting for nodes who opt for checkpointing instead of continuous history is to download a checkpoint per 2016 blocks or 7 days. As another example, one upcoming implementation involves downloading a checkpoint and then some preceding history; if invalid blocks are found in history, then the compromised checkpoint is eschewed altogether.
- If a state-attack succeeds, then the nodes that had longer histories than the state-attack rollback are immune to any balance alteration.
- If one falls to a state-attack, the worst case is that he would simply need to re-download a longer segment of the PascalCoin blockchain history and recover his balance(s). The state-attacker would be stuck with invalid balances unless he continues to fool others.

Technically, no blockchain data in PascalCoin is ever deleted even in the presence of the SafeBox. Since the SafeBox is cryptographically equivalent to a node with the entire history, PascalCoin nodes are not expected to contain infinite history. But for any reason(s) one may have, one could still keep all or some of the PascalCoin blockchain history as an option.

The SafeBox security model actually offers an advantage compared to the full blockchain model in terms of security because it places far less dependence on the full blockchain history (i.e. archival nodes). In the long term, the full blockchain model would inevitably lead to some form of centralization as well as a potential security risk. As a result, the SafeBox security model is at least on par with the full blockchain security model on the macro level.

On the micro level, the SafeBox security model is barely diminished to that of the full blockchain model. However, it is still far greater than the simple payment verification (SPV) security model. In terms of cost-benefit analysis, the SafeBox is superior to full blockchain because the SafeBox needs only "1%" of storage yet retains "99%" of security.

Thanks to the SafeBox's design, the user would have full assurance that he is on the longest Proof-of-Work chain without needing to download the entire blockchain history – a unique innovation that offers many advantages.

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."

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 is 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. This type of double-spend – different from the double-spend scenario based on majority hashpower – is virtually impossible in PascalCoin.

The buyer's transaction to the merchant is simply a delta-operation to debit/credit a quantity from/to accounts. 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. A double-spend attempt would have the same operation identification as the original transaction which means that it is easy to detect a double-spend. Therefore, if the buyer tries to double-spend the coffee funds after receiving the coffee, the double-spend transaction will not propagate the network since nodes do not propagate a transaction if it double-spends a current pending transaction. All PascalCoin nodes – including those that adopt the lightweight SafeBox model instead of the full history – can easily detect double-spends while doing so in a UTXO model is relatively inefficient and complex.

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 Pascal payment. If the service replies "No double spend detected" then it is essentially impossible for a double-spend to occur since the majority of nodes are honest and will not propagate a double-spend attempt. Such a service would prevent an attacker from launching a double-spend from both sides of the globe which otherwise would may have possibly resulted into half of the network carrying the valid transaction and the other half of the network carrying the double-spend. The only way an attacker would be able to overcome this additional security measure would be if he colludes with a malicious miner – one that wins the next block after the original transaction – with regards to a private transaction. This is a costly and a probabilistically unlikely proposition.

A merchant node will also be able to enable 0-confirmation insurance on payments. When a node receives a payment, it then takes out an instant insurance on that payment via a monetized API invocation to a smart contract. If that payment turns out to be invalid, the smart contract automatically compensates the merchant account for the amount of the transaction. The insurance fee is paid by the merchant on the insurance request – no setup, registration, contracts or loss of privacy are required.

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 are practically as good - sufficient for micropayments and everyday commerce.

ZERO-FEE TRANSACTIONS

Another unique feature of PascalCoin is that users are allowed to make one free transaction per block (i.e. every 5 minutes). This gives users 288 free transactions per day, a very reasonable number. The consensus rules for this are simply that a public key can have a maximum of one zero-fee transaction in the memory-pool and/or block at any time. This simple approach solves the transaction spam problem whilst providing users a very reasonable number of free transactions per day. Should one transact more than once in 5 minutes, the very minimum PASC transaction amount (currently set at 0.0001 PASC) will be attached as a fee. Should the price of PASC rises, another zero will be added to the minimum PASC transaction amount via a Protocol Improvement Proposal (PIP).

Zero-fee transactions are achieved on the merits of PascalCoin's extremely high throughput in which the fee bidding competition of transaction priority typical in other UTXO cryptocurrencies is moot. The fee pressure only starts rising as current throughput approaches maximum throughput which may not occur even if the PascalCoin protocol is mass adopted. The fact that there is no fee incentive needed to run a node, because of its extremely lightweight overhead, beyond the consensus rewards also facilitates zero-fee transactions.

Zero-fee transactions apply only to "vanilla" transactions, namely those that are not private and do not involve smart contracts. Transactions involving privacy and/or smart contracts will involve negligible fees.

ACCOUNT NUMBERS, NAMES AND TYPES

PascalCoin facilitates value-transfer between users by allowing them to transact funds (Pascal or 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.

One of the key features of PascalCoin is that accounts can have unique names which are publicly 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 serve a fundamental purpose in Layer-2 applications and Monetized API. 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 cases. For example, browsing accounts with type = 2 could be like browsing a list of chat rooms. How users interact with such Layer-2 applications via Monetized API will be described more in this whitepaper.

SMART CONTRACTS

PascalCoin offers powerful approaches to smart contracts due to its SafeBox model. The roadmap includes a Layer-2 overlay network for full Ethereum-style contracts. In particular, smart contracts in PascalCoin are isolated form the Layer-1 financial network and operate independently on Layer-2; in other words, Layer-1 will be used as a financial settlement network for Layer-2 overlay network. The characteristics of such an architecture are as follow:

- Running a smart contract engine like the Ethereum Virtual Machine (EVM) over PascalCoin would be possible by maintaining a side-chain pinned to a PASA account. Each PASA has infinite address space for its sub-addresses (unlike PASAs, infinite sub-addresses do not affect the SafeBox), meaning that side-chain users do not need to own a PASA.
- Since side-chains are pinned to PASAs, they are intrinsically sharded. Inter-shard communication would simply be transactions between PASAs. Gone are the complications that accompany the traditional sharding approach such as used in Ethereum.
- Layer-2 side-chains will provide very strong anonymity since funds are all pooled and keys are not used to unlock them.
- The network would not be impacted by the large volume of transactions since the natural process of checkpointing discards these transactions after 100 blocks.
- Smart contracts can be developed in any language and target any platform, since smart contract execution and consensus are separate concerns.
- Layer-2 consensus can adopt multiple forms such as Delegated Proof-of-Stake (DPoS) and Directed Acyclic Graph (DAG) depending on the side-chain's goals.

The infinite scalability of PascalCoin's Layer-1 will extend to PascalCoin's Layer-2 as well. This Layer-2 architecture is designed in which the computation is separate from consensus, in effect removing any bottleneck. To be precise, the speed and scalability of Layer-2 smart contracts are fully independent of PascalCoin's network. Horizontal scaling also exists in this paradigm as there is no interdependence between smart contracts and states are not managed by side-chains. One would be able to run the entire global financial system on PascalCoin's smart contract platform and it would scale infinitely and securely.

Implementing this infinitely scalable smart contract platform as a Layer-2 architecture is the focus of version 5 and a whitepaper add-on for this development will be released shortly.

ANONYMITY

Due to PascalCoin's design, transactions to and from accounts can be easily audited by anyone. For many use cases, this is an excellent feature since it makes auditing, reconciliation and accounting very simple. However, PascalCoin is also committed to providing equally anonymizing features for those use cases where privacy is important. As a result, PascalCoin has a strong privacy roadmap that has already been partially rolled out.

In PascalCoin version 1, users could transfer Pascals privately using a PASA-exchanging approach. Instead of sending Pascals to each other's accounts, the account itself would swap ownership. So as long as the key exchange was transmitted through secure channels, it would not be possible to trace this transaction.

PascalCoin's version 3 added an in-protocol PASA and Pascal tumbling capability. Users who elect for privacy would only need to hit a checkbox, pay a marginal fee (currently set at 0.0001 Pascal), and click send. Under the hood, the wallet will automatically negotiate with other nodes for a set of obfuscating PASA/PASC swaps involving many accounts and merge these all into a single multi-operation. This process can then be chained several times offering exponentially higher security with every link, rendering the original transaction virtually impossible to decipher.

In addition, nodes will be able to earn fees by offering their latent PASC and PASA for these tumbling operations whilst simultaneously providing a rich set of decentralized PASC/PASA to participate in the tumbling. This would result in a fast, cheap, seamless and genuine anonymity for PascalCoin users.

The PascalCoin Foundation is also sponsoring R&D proposals for adding zk-SNARKs (or a similar, suitable approach) into the protocol for private balances. It is also investigating smart contract based privacy approaches.

MONETIZED API

Due to reliable 0-confirmation transactions, PascalCoin allows a new form of decentralized application coined here as " Monetized API"[5]. 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.
Some examples of applications of Monetized API are as follow.

EXAMPLES:

Pascal Chat: An account can serve as a chat room where 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 decentralized 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 whose shopping cart is reduced to a single "Payload Code." The buyer merely sends a Pascal 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 are 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 Pascals or more. When a transaction is sent to 77-44 containing 10 Pascals or more, its payload is examined for the presence of a public key. If nothing is found, the transaction is discarded. If a public key is found, the service then finds a free account in its 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.

DATA TRANSFER

PascalCoin's SafeBox architecture allows accounts to transfer data between themselves in a secure and private manner. This is possible due to the ability to attach a 256-byte payload to each transaction. Via this mechanism, a user can split any file into multiple segments and transfer each segment as an operation to the receiver. This data can be ECIES encrypted so that only the receiving account can reconstruct the original file/audio/video. It can also be AES encrypted so that bearers of the shared secret can reconstruct the file. Alternatively, the data transfer could be public and available for everyone. Since PascalCoin's nodes need to keep only 100 blocks, blockchain-based data exchanges do not result in a blockchain bloat. This Layer-1 feature can be leveraged by Layer-2 smart contracts to provide infinitely scalable, global Data Storage Networks (DSN) and even decentralized document management solutions for both private and public data.

SELF-FUNDED AND SELF-GOVERNED COMMUNITY

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 development team, led by Albert Molina, are mature, exceptional developers with proven track records of delivering high-quality software.

The developer fund exists as a percentage of mining rewards as voted by the PascalCoin community. This developer fund is set as 20% of the mining rewards and is governed by a decentralized autonomous organization (DAO) facilitated by the PascalCoin Foundation. This DAO will eventually be transformed into an autonomous smart contract platform. Not only are the developer fund disbursements voted upon the community, but PascalCoin's development roadmap is also voted upon the community via the Protocol Improvement Proposals (PIPs). This decentralized governance also serves an important benefit as a powerful deterrent to fork wars that befall many cryptocurrencies.

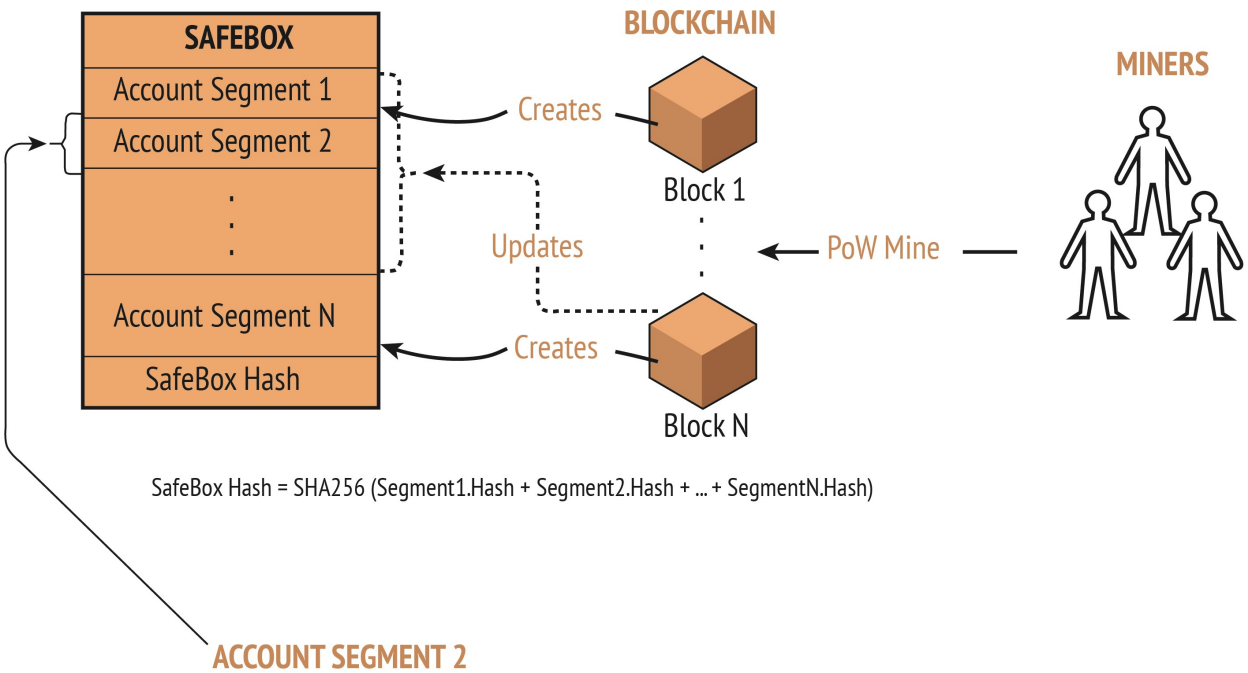
WRITTEN IN FREE PASCAL

PascalCoin was designed and written from scratch in the Free Pascal programming language without copying a single line of code from any other project. 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 its modern advances and upgrades, has become a great language for writing high- performance, cross-platform native code. While the Free Pascal programming language is used for PascalCoin's core development, dapps and any external implementations based on PascalCoin can be written in any language. One advantage using the Free Pascal programming language is that it raises the barrier higher for forking a clone cryptocurrency based on PascalCoin. It would be almost impossible to fork PascalCoin unnoticed due to its Pascal-based codebase unless one masters the Pascal programming language then rewrite the entire codebase into a different language.

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.



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 SAFEBOX

Before the first block is created, the very first SafeBox 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 an Account Segment containing N new accounts, where N is defined by the protocol (N is set to 5 in 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

Similar 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 its name or to mark it for sale, etc. The most important and common operation is the Transaction operation 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 blockchain that modified this account. This value helps to identify a stale, unused account.

ELEMENT NAME	DATA TYPE	DESCRIPTION
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 contents 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.

ELEMENT NAME	DATA TYPE	DESCRIPTION
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 an account 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 accounts 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 lists its public key which will be assigned to the N new accounts created in the Block Account.
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 to 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 is used for the Proof-off-Work.

CHECKPOINT TORRENTING

In upcoming release(s), 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.

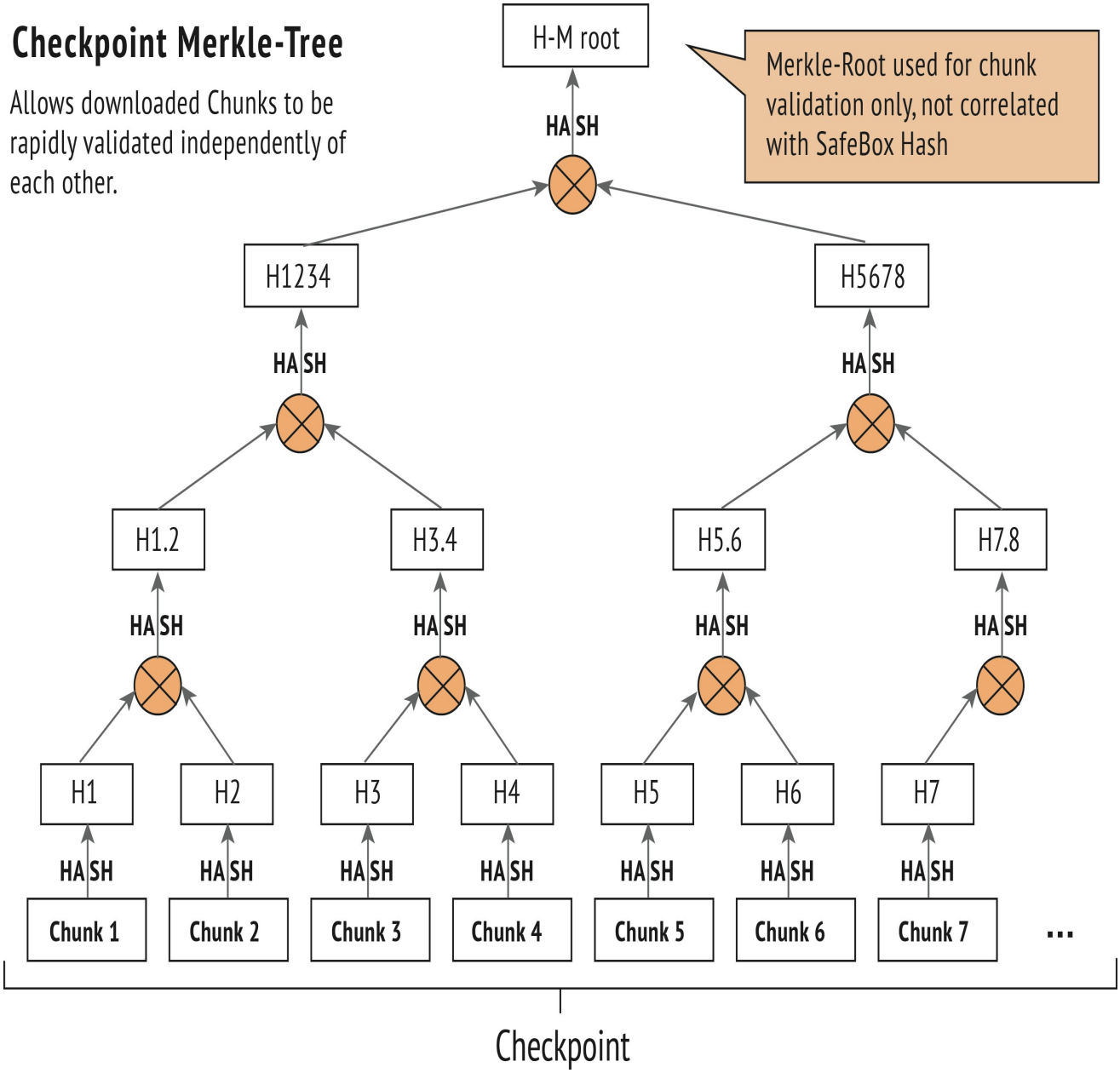
During transmission, chunks are significantly compressed which greatly improves synchronization speed and reduces network traffic. Chunks are then independently validated using the provided merkle-tree (see diagram below). Once all chunks are downloaded, the checkpoint is reconstructed and verified for 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.

The next page contains the diagram of the checkpoint merkle-tree.

Checkpoint Merkle-Tree

Allows downloaded Chunks to be rapidly validated independently of each other.



PASA DISTRIBUTION MODELS

COMMODITIZATION BY DESIGN

PASAs need to be commoditized by the SafeBox's design, meaning that PASAs cannot be obtained at no charge to prevent systematic abuse. This raises two issues that are tackled appropriately in PASA distribution models:

- As a chicken and egg problem, how would one purchase a PASA using Pascal in the first place if one cannot obtain Pascal without a PASA?
- How would the price of PASAs stay low and affordable in the face of significant demand?

With regards to the chicken and egg problem, there are many methods – some finished and some unfinished – to obtain a first PASA which may be free depending on the method used.

It is worth noting that while PASAs need to be commoditized to a degree as part of the SafeBox's architecture, they will always remain affordable at a very inexpensive price as well as continually become easier to acquire over time as more acquisition methods are implemented.

FIRST PASA ACQUISTION METHODS

Free first PASA acquisition methods:

- Submit a SMS message through FreePasa.org and a free PASA from PascalCoin Foundation will automatically be granted to a public key
- Obtain a free PASA from PascalCoin Foundation by simply activating your Android/iOS wallet using a SMS number
- Send a PascalCoin related tweet reaching at least 10 people and a free PASA will automatically be given from the PascalCoin Foundation
- Obtain unlimited pseudo-PASAs in version 5 with a Layer-2 implementation

Minimal charge for first PASA acquisition methods:

- Purchase a PASA at an inexpensive price from:
  - GetPasa.com
  - PascWallet.com
- Obtain PASAs via mining rewards (currently set as 5 PASAs per 50 PASC)
- Purchase a PASA OTC from someone

PASA ECONOMICS

As for ensuring the PASA market at large remains inexpensive and affordable, this would be achieved in two ways:

- Decentralized governance of the PASA economics
- Unlimited and free pseudo-PASAs based on Layer-2 in the next version release which would reduce demand for Layer-1 PASAs

The decentralized governance of PASA economics would work as follows. If the price of a PASA increases then there are many available options via decentralized, community-driven PIPs to keep the PASA price low, including but not limited to:

- Increasing the supply of PASAs mined per PASC (currently set at 5 PASA per 50 PASC)
- Issuing accounts per block in accordance with Moore's Law
- Dynamic and instant account creation using new and yet-to-be-implemented Create Account & Destroy Account operations
- One-time issuance of millions of new accounts to entities (PascalCoin Foundation, PascWallet.com, etc.) for secure, decentralized distribution methodologies

ACCOUNT RECOVERY

Coin rot is an economic issue afflicting almost all other cryptocurrencies. Coin rot is the phenomenon of coins being permanently lost due to lost/corrupted keys and/or natural death. It is estimated that 20% of Bitcoins have already rotted. Coin rot as applied to PascalCoin, especially in the context of PASA, poses a minor but unnecessary weight on the SafeBox's architecture. As a result, account recovery will be executed after 4 years of full inactivity (i.e. no single transaction from the PASA of interest) and the contents (both PASC and PASA) would be distributed to miners. However, this 4-year period will likely be extended to a 10-year period with an upcoming PIP.

RANDOM HASH ALGORITHM

SPECIFICATION

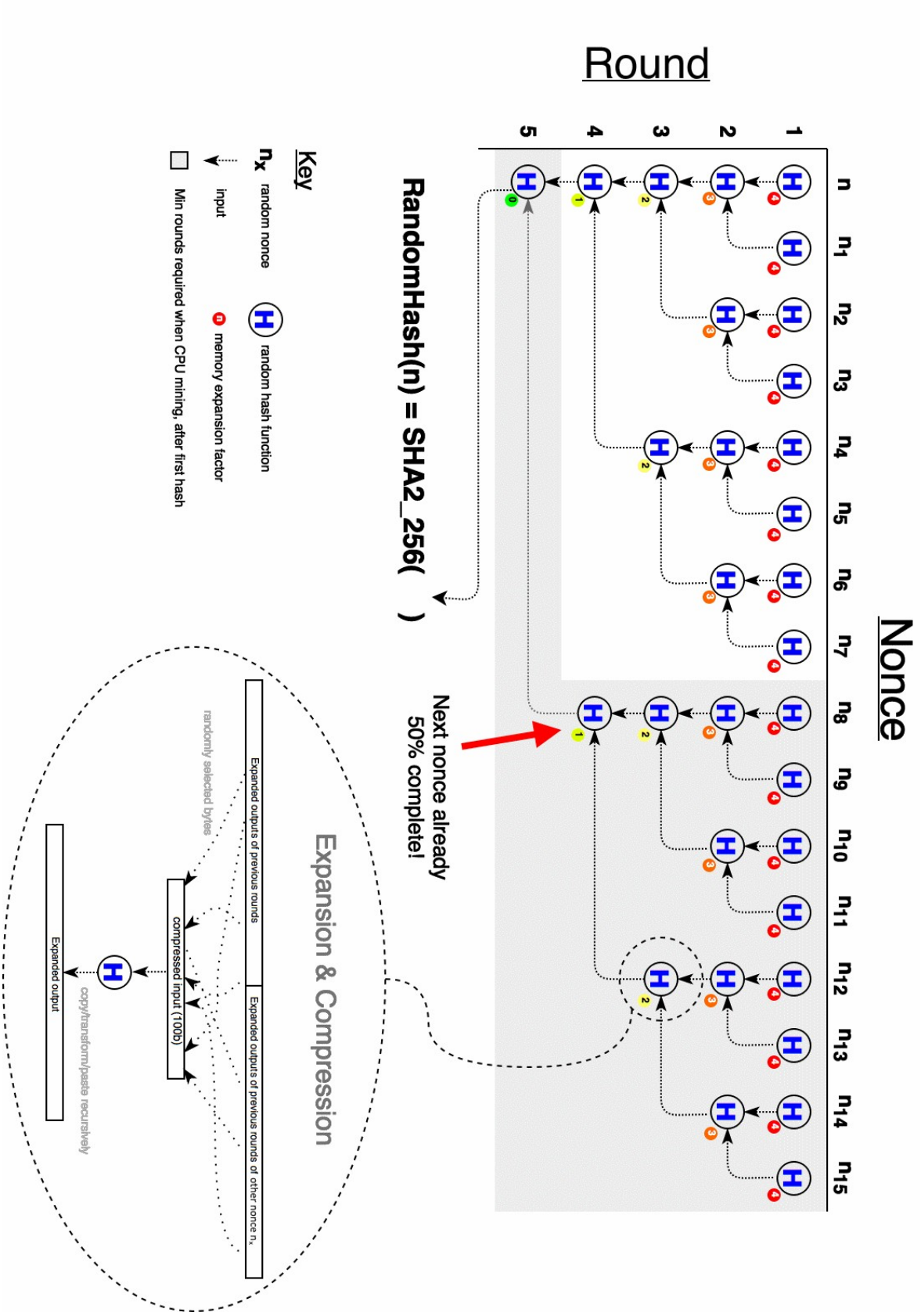
A low-memory, GPU and ASIC-resistant hash algorithm called Random Hash is proposed to resolve and prevent dual-mining and mining centralization. Random Hash, defined first here, is a "high-level cryptographic hash" algorithm that combines other well-known hash primitives in a highly serial manner. The distinguishing feature is that calculations for a nonce are dependent on partial calculations of other nonces, selected at random. This allows a serial hasher (CPU) to re-use these partial calculations in subsequent mining saving 50% or more of the work-load. Parallel hashers (GPU) cannot benefit from this optimization since the optimal nonce-set cannot be pre-calculated as it is determined on- the-fly. As a result, parallel hashers (GPU) are required to perform the full workload for every nonce. Also, the algorithm results in 10x memory bloat for a parallel implementation. In addition to its serial nature, it is branch- heavy and recursive making in optimal for CPU-only mining.

Within the scope of changing the hash algorithm, other hash algorithms were considered like Equihash. However, these were ruled out due to their excessive memory consumption contradicting PascalCoin's vision of globally decentralized network that runs fine on low-end hardware available anywhere on this world. Requiring voluminous amounts of fast memory to validate blocks is not consistent with this vision.

OVERVIEW

1. Hashing a nonce requires N iterations (called rounds)
2. Each round selects a random hash function from a set of 18 well-known hash algorithms
3. The input at round x is salted with the outputs of all prior rounds
4. The input at round x is salted with the output of all prior rounds of a different nonce,randomly determined
5. The input at round x is a compression of the transitive closure of prior/neighbouring round outputs to the size of 100 bytes
6. The output of every round is expanded for memory-hardness
7. Randomness is generated using Mersenne Twister algorithm
8. Randomness is seeded via MurMur3 checksum of current round
9. The final round is then hashed again via SHA2_256, in keeping with traditional cryptocurrency approaches

RANDOM HASH DESIGN



RANDOM HASH ANALYSIS

CPU Bias

The Random Hash algorithm is inherently biased towards CPU mining due to its highly serial nature. In addition, Random Hash allows CPU miners to cache the partial calculations of the other nonces and resume them later. This allows CPU miners to save 50% of the work during mining. This is formally proven in a separate Random Hash whitepaper, but is easy to grasp as follows - in order to complete a nonce to round N, another nonce needed to be completed to round N-1. The other nonce requires 1 more round to complete, saving 50% of the work. This optimal nonce-set cannot be pre-calculated, and can only be enumerated. As a result, serial mining (CPU) does 50% the work of parallel mining (GPU).

Memory Complexity

Random Hash is memory-light in order to support low-end hardware. A CPU will only need 5MB of memory to verify a hash. During mining, it will need 10MB per thread (when utilizing the 50% bias mentioned above) – an easy requirement. It's important to note that Random Hash consumes most of the memory in the initial rounds and little in the final rounds. This is deliberate in order to hinder GPU mining. For example, suppose a GPU has 5GB of memory. A naive hasher could attempt to batch 1000 nonces for parallel evaluation (since each nonce requires 5MB). However, since each nonce depends on 15 other nonces and most of the "5MB per nonce" is consumed in the early rounds of those nonce evaluations, the GPU will run out of memory quickly. The batch size needs to be divided by 15 in order to utilize the 5GB effectively, which means most of the GPU memory is wasted on partial nonce calculations from the early rounds. In that scenario, the GPU can only effectively compute 20 nonces per 1GB of memory. A CPU can easily compete with this and implement intelligent parallel mining by using other threads to mine the less-partially calculated nonces. This could potentially give a CPU significantly greater than 50% advantage, but this approach needs further research.

GPU Resistance

GPU performance is generally driven by parallel execution of identical non-branching code-blocks across private regions of memory. Due to the inter-dependence between hashing rounds, the slower global memory will need to be used. Also, due to the highly serial nature of Random Hash's algorithm, GPU implementations will be inherently inefficient. In addition, the use of Mersenne Twister to generate random numbers and the use of recursion will result in executive decision making further degrading GPU performance. Most importantly, since nonces are interdependent on other random nonces, attempts to buffer many nonces for batch hashing will result in high memory wastage and 200% more work than a CPU. This occurs because each buffered nonce will require calculation of many other unbuffered dependent nonces, rapidly consuming the available memory. A CPU implementation does not suffer this since the optimal nonce-set to mine is enumerated on-the-fly as each nonce completes. Another important feature is the pattern of memory expansion factors chosen for each round. These were deliberately chosen to hinder GPUs by amplifying the memory needed for their wasted calculations.

As a result, it's expected that GPU performance will at best never exceed CPU performance or at worst perform only linearly better (not exponentially as is the case now with SHA2-256D).

ASIC Resistance

ASIC-resistance is fundamentally achieved on an economic basis. Due to the use of 18 sub-hash algorithms, it is expected that the R&D costs of a Random Hash ASIC will mirror that of building 18 independent ASICs. This moves the economic viability goal-posts away by an order of magnitude. For as long as the costs of general ASIC development remain in relative parity to the costs of consumer grade CPUs as of today, a Random Hash ASIC will always remain "not worth it" for a "rational economic actor."

Furthermore, Random Hash offers a wide ASIC-breaking attack surface. This is due to its branch-heavy, serial, recursive nature and heavy dependence on sub-algorithms. By making minor tweaks to the high-level algorithm, or changing a sub-algorithm, an ASIC design can be mostly invalidated and sent back the drawing board.

This is true since ASIC designs tend to mirror the assembly structure of an algorithm rather than the high-level algorithm itself. Thus by making relatively minor tweaks at the high-level that necessarily result in significant low-level assembly restructuring, an ASIC design is made obsolete. So long as this "tweak-to-break-ASIC" policy is maintained by the PascalCoin developers and community in addition to the economic impracticality of building a Random Hash ASIC, ASIC resistance is guaranteed.

Random Hash Variations

Variations of Random Hash can be made by varying N (the number of rounds required) and M (the memory expansion). For non-blockchain applications, the dependence on other nonces can be removed, providing a cryptographically secure general-purpose,albeit slow, secure hasher.

It is also possible to change the dependence graph between nonces for stronger CPU bias. For example, requiring the lower rounds to depend on more than one nonce and the upper rounds on no nonces at all may allow further CPU vs GPU optimization, and similarly for memory expansion factors.

Extra, albeit unnecessary, strengthening can be added in the initial rounds of hashing by using the hash of the block header for seeding instead of the block header itself. In the analysis of the author, this is unnecessary and has subsequently been removed.

Random Hash Whitepaper

There is a separate Random Hash whitepaper written that goes into greater details on the algorithm. This whitepaper provides pseudo-codes and reference implementation codes for the algorithm as well as formal proofs.

ROAD MAP

Accomplished:

[✓] Initial Main Net Launch
[✓] Miner Infrastructure
[✓] Checkpointing
[✓] Infinite Scaling
[✓] Random Hash Algorithm
[✓] Layer-2 Operation Support
[✓] In-Protocol PASA Exchange
[✓] In-Protocol PASC & PASA Mixer
[✓] Account Names & Types
[✓] 50% Inflation Reduction
[✓] PascalCoin Foundation Established
[✓] Decentralized Development Process (PIP)
[✓] Blockchain Voting Website
[✓] SafeBox Voting (Phase 1)
[✓] Add 20% Mining Reward for Developers
[✓] 90% Developer Reward Open to Community
[✓] Community- Run Mining Pool
[✓] RHMiner Optimizations

Goals:

[ ] New Wallet (with OS X support).
[ ] iOS & Android Wallet
[ ] Mobile/Light-Client Support
[ ] Translation Portal & Full Translations.
[ ] Infrastructure for DAO
[ ] SafeBox Voting (phase 2)
[ ] Expand PASA Acquisition Methods
[ ] Checkpoint Torrenting
[ ] Account Data
[ ] Super-Optimization Phase (goal: 100k/sec)
[ ] Double Spend Detection & 0-Confirmation Insurance
[ ] Schnorr multi-sig (mostly completed)
[ ] Private Balances & Payments (zk-SNARKs)
[ ] Full C# Node Implementation
[ ] Multi-platform TipBot

[ ] Layer-2: PascalDex (decentralized exchange)
[ ] Layer-2: Overlay Network
[ ] Layer-2: Infinite Address-Space
[ ] Layer-2: PascalMessenger (PASC MSG)
[ ] Layer-2: Accounts with Private Balances
[ ] Layer-2: PASC Data Storage Network (DSN)
[ ] Layer-2: Private Send Protocol
[ ] Layer-2: PascalTokens

ACKNOWLEDGEMENTS

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

REFERENCES

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

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

[3] Diagram of Monetized API