Updated: September 2019, Version: 5.0

Pascal: An Infinitely Scalable Cryptocurrency

Herman Schoenfeld herman@pascalcoin.org

Albert Molina albert@pascalcoin.org

Pascal is a “next-generation” cryptocurrency designed on an “infinitely scalable” architecture. It is based on a simplified accounts model rather than a UTXO-model and designed for ultra-high transaction throughput requiring only constant storage on nodes. It achieves this by introducing a new cryptographic database known as the SafeBox that works in tandem with a proof-of-work blockchain (which can be deleted). In addition to the payments use-case, Pascal provides a scalable consensus and financial layer for Layer-2 applications intended as the primary real-world use-case for Pascal.

INTRODUCTION

Pascal is a next-generation cryptocurrency that solves many of the limitations of existing cryptocurrencies. Its primary focus is on solving the "scalability issue." Stress-test results of the testnet 4.1 codebase shows the mainnet capacity can achieve 1,600 transactions per second.

Secondly, Pascal solves the long-term blockchain "storage issue" by introducing deletable blockchain technology. This technology allows Pascal nodes to securely delete the blockchain beyond the last 100 blocks. By storing the flow of transactions rather than the infinite history, the Pascal network is the first (and only known) cryptocurrency that can theoretically run for infinite time at maximum throughput while requiring only constant storage. In this manner, Pascal achieves "infinite scalability".

Lastly, Pascal's account-based model greatly simplifies the ledger structure and provides users a familiar "bank account" experience whilst remaining decentralized and permissionless like Bitcoin. This simplified architecture also enables Layer-2 protocols to be enveloped within Pascal transactions thus facilitating a new type of decentralized application architecture (dApps). Pascal dApps are intended to run independently from each other and use Pascal for periodic financial settlements and consensus. In this manner, a naturally sharded Layer-2 dApp runtime emerges that allows dApps to scale, store, execute and process their data and logic independently from each other without impacting each other's performance (unlike existing platforms which process all dApp business logic at Layer-1).

Currently, the Pascal network provides the following features:

- Fair economic distribution (no pre-mine or ICO)
- 5 minute block-time, 2 year halving epoch, 42 million coin issuance and 1 coin per block tail-emission
- Near-instant, zero-fee transactions
- Secure 0-confirmation transactions
- Simple account-based model using human readable and writeable account number addresses
- In-protocol account exchange and atomic-swap capability
- Transaction throughput of 1600 Transactions Per Second (TPS)
- Deletable blockchain system
- Cryptographically secure account histories (a DAG structure)
- Low-memory ASIC & GPU resistant Proof-of-Work algorithm
- Layer-2 extension points for a wide domain of applicability.

On the roadmap, Pascal aims to provide:

- Layer-1 Protocol Governance at a block-by-block granularity
- Layer-2 Proof-of-Stake Overlay Integration
- 100k+ transactions per second (and beyond).

CURRENT HISTORY

Pascal was launched originally by Albert Molina[16] in 2016 as Pascal **version 1**. It was launched in the same way as Bitcoin as a "fair-launch" coin without any ICO or pre-mine. This initial implementation introduced a new cryptographic data structure known as the SafeBox. The blockchain and the SafeBox work in tandem to maintain the balance of users funds and data. The SafeBox functions as a "ledger balance" that maintains the up to date state of all user accounts whereas the blockchain functions as the "ledger" containing the transactions which mutated those user accounts. Whilst the initial version of Pascal allowed pruning of the blockchain, a node was required to download and verify the full chain at least once.

This limitation was overcome in **Version 2** when Herman Schoenfeld joined the project and provided a solution that enabled nodes (and the network itself) to delete the blockchain whilst retaining its aggregated proof-of-work security[14]. Only the SafeBox and last 100 blocks would ever be required to fully participate in the network. Version 2 also introduced in-protocol account exchanging, account names and difficulty adjustment algorithm changes.

Version 3 introduced significant new changes including a 50% inflation schedule reduction[1], 20% developer reward[2] to fund development and infrastructure, batch transactions[3] and a new GPU/ASIC resistant proof-of-work algorithm called RandomHash[5].

Version 4 introduced layer-2 data transactions[5], fast block propagation[6] and significant performance enhancements.

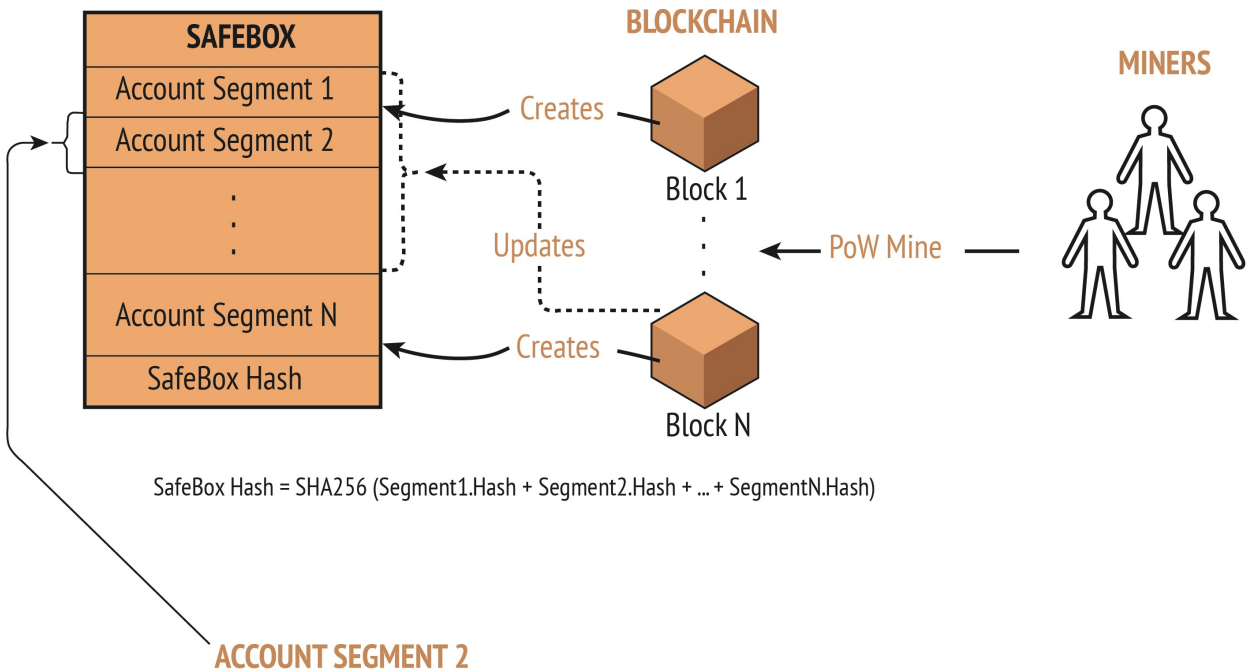
Version 5 introduces cryptographically secure account histories[7], deletable SafeBox and light-nodes[8], layer-2 addressing format[9], new mobile wallets and atomic swaps. Also, Pascal is being lightly rebranded from PascalCoin to "Pascal" in this version.

SELF-GOVERNED COMMUNITY

In version 3, a 20% block reward was added into the protocol to fund further development of Pascal and its ecosystem. This funding belongs to the community, for the community and appropriated by the community using blockchain-voting methodologies (and social media platforms).

An interim foundation was created to administer these funds called PascalCoin Foundation Limited, registered in Australia. The foundation is only a temporary custodian of the protocol-generated funding. Long-term, this funding will be managed by a Layer-2 dApp implementing a Layer-2 protocol that is secured by a proof-of-stake overlay network. This protocol will appropriate funding based on Layer-1 voting generated by the Pascal users themselves.

LAYER-1 ARCHITECTURE



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)

SAFEBOX

The SafeBox is a cryptographic data-structure invented by Albert Molina in 2016 and improved by Herman Schoenfeld since. SafeBox offers an alternative approach to decentralized consensus ledgers that rival traditional Blockchain and DAG based approaches. The main differentiating factor is its simplicity.

The purpose of the SafeBox is to store user account balances independently from the blockchain. If the blockchain is the “ledger”, the SafeBox is the “ledger balance”. The primary innovation of the SafeBox is its cryptographically secure connection to the blockchain. By examining the SafeBox alone, a user can calculate the “aggregated proof-of-work” required to have constructed that SafeBox without needing the blockchain to verify. This allows the network to safely delete the blockchain over time yet retain the historical proof-of-work provided by that blockchain.

Structurally, the SafeBox is composed of a set of Account Segments. Each account segment is a set of Accounts. Accounts are atomic records that maintain user’s balance, public key and other data. When the network mines a new block, the block commits to the current SafeBox and appends a new Account Segment.

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 Pascal block does not directly reference the previous block.

Instead it references the previous SafeBox state which is compactified into a hash called the "SafeBox Hash". In version 5, it is called the SafeBox Root[8] and is calculated as the Merkle-Root of the Account Segments. When a Block is "applied" to the SafeBox, it appends a new Account Segment containing new Account's (belonging to the miner).

OPERATIONS

Similar to Bitcoin and other cryptocurrencies, blocks in Pascal are containers for transactions referred to as "operations." They are called operations because they are generalized transactions that can perform much more than simple value transfers between accounts. For example, there are operations to change an account's key, change its unique name, mark it for sale, perform atomic swaps, send data and batch combinations thereof. The most important and common operation is the Transaction operation which transfers funds between accounts.

ACCOUNT SEGMENTS

The SafeBox is basically the list of all users Accounts. However, these Accounts are grouped into segments called an Account Segments. An Account Segment is generated every time a miner appends a block to the SafeBox. In other words, the SafeBox is extended atomically with a new Account Segment each time a new block is mined.

Currently, an Account Segment only contains 5 accounts, but this will become a protocol variable[10] in coming protocol upgrades. The SafeBox Hash (or SafeBox Root[8]) is basically an aggregated hash of all its containing Account Segments. New blocks must commit to the SafeBox by referencing the SafeBox Root in the block header.

ACCOUNTS

Accounts are the atoms of the SafeBox and the digital property of users. Accounts are initially awarded to the miners who then distribute them to users via a market mechanism. A Pascal Account is also known as a PASA. A PASA contains a variety of fields including its balance (PASC), public key, unique name, data fields and other internal state fields used for atomic swaps and in-protocol exchanging.

INFINITE SCALING

In the context of this document, "Infinite Scaling" is defined as *"the ability of a network to achieve an infinite running time at maximum throughput using constant storage"*. In other words, it's the ability of a network to run forever, at maximum speed, without needing more and more storage devices to support it. Pascal is the only (known) cryptocurrency that achieves Infinite Scaling. All other cryptocurrencies, blockchains and DAGs depend on their full and permanently growing histories. In the view of the author, such a dependency is a "genetic defect" of such architectures as it imposes a maximum viable age to their networks, in terms of usability.

For example, when the size of a blockchain becomes unmanageable, new nodes cannot join the network faster than

it grows and existing nodes struggle to keep up and abandon the network. This results in the rapid centralization of the network resulting in a total compromisation of its security.

This issue is already emerging in other projects that were not founded on an Infinitely Scalable architecture. Such networks struggle to support their (relatively low) demand and often experience network outages, elevated orphan rates and the occasional (unintentional) splitting of their network topology. Eventually, these networks will become unusable entirely.

It's clear that the long-term viability of a cryptocurrency network depends on an "Infinitely Scalable" architecture, something Pascal has pioneered (exclusively). In a global adoption scenario where user demand is orders of magnitude higher than the current cryptocurrency market an Infinitely Scalable architecture will become mandatory.

HOW DOES PASCAL ACHIEVE "INFINITE SCALING"?

Simply put, since the SafeBox contains the account balances, the blocks in Pascal are deleted past a height of 100 (the checkpoint). 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 are ever required at any time (and the SafeBox).

A checkpoint is simply the 100th block from the current tip block. So if the tip block is 200 then the checkpoint block is 100. If the tip block is 201, the checkpoint block is 101. When a new node joins the network, it only needs the latest SafeBox and the last blocks from the checkpoint to the tip.

A node can always choose the correct SafeBox by selecting (and verifying) the one with most aggregated proof-of-work. The ability to verify the aggregated proof-of-work of the entire blockchain without needing the blockchain is the primary innovation of Pascal and the reason why the Pascal blockchain can be deleted.

TRANSACTION THROUGHPUT

Pascal currently achieves 1600 transactions per second on the production network. Compared to Bitcoin's 5 TPS and Ethereum's 15 TPS, Pascal has already innovated in this area by several orders of magnitude. The original goal for Pascal was to achieve VISA-level scalability. The VISA network performs 1700 TPS, on average. At peak, this number can burst anywhere up to 70,000 TPS.

It is an empirical fact that Pascal has already achieved VISA-level scalability (on average).

Whilst other projects make similar (or even grander) claims, they have not demonstrated this capability on their production networks. Also, their dependence on LevelDB casts doubts on their claims since LevelDB can only achieve 300 TPS. Pascal uses its own custom storage database developed by Albert Molina which is built-for-purpose resulting in superior performance.

Going forward, the Pascal roadmap includes fundamental enhancements that can increase this throughput to 100,000 TPS and beyond. Since the organic demand for the Pascal network is far below what it can support today, the developers reserve the right to disclose this technology at a suitable time so as to minimize technology leaks to other projects.

RANDOMHASH2: GPU & ASIC-RESISTANT PROOF-OF-WORK

Pascal's blockchain is secured by a low-memory, GPU and ASIC resistant hash algorithm called RandomHash2[11]. RandomHash2 is a "high-level" hash algorithm that combines 77 well-known cryptographic hash algorithms in a random, non-deterministic manner. The algorithm is low-memory, highly serial, recursive, branch-heavy and involves extensive executive decision-making. As a result, it is intrinsically inefficient for GPU-based processing and, due to its complexity, not economically viable for an ASIC implementation.

The primary innovation of RandomHash2 (and RandomHash[4]) is how it is used in the mining process. The algorithm is designed such that the evaluation of a nonce is dependent on the partial evaluation of random, neighboring nonces. By injecting this nonce-interdependency into the algorithm, RandomHash2 allows a serial miner (CPU) to mine significantly faster than a parallel hasher (GPU) since the optimal nonce-set can only be enumerated on-the-fly, not pre-ranged for parallel evaluation. As a result, parallel hashers (GPU) are required to perform the full workload for all rounds whereas serial hashers (CPU) can re-use the partially completed nonce calculations from previous rounds. In RandomHash2, this "CPU bias" optimization gives CPUs a 500% higher hashrate.

Algorithm Overview:

1. Hashing a nonce requires N iterations (called levels), which are evaluated recursively;
2. N varies per nonce in a non-deterministic manner between MIN_N=2 and MAX_N=4, inclusive;
3. Each level in (1) also depends on J neighboring nonces, determined randomly and non-deterministically;
4. The value J is restricted to MIN_J=0 and MAX_J=4;
5. Each level selects a random hash function from a set of 77 well-known cryptographic hash algorithms;
6. The input digest hashed at a level is the compression of the transitive closure of the hash outputs of all it's prior-levels (1) and it's neighboring nonce's prior-levels (3);
7. The output of every level is expanded for (low) memory-hardness;
8. Randomness is generated using Mersenne Twister algorithm;
9. Randomness is always seeded using last DWORD of a cryptographic hash algorithm output;
10. The first input is pre-hashed as using SHA2-256;
11. The last output is post-hashed using SHA2-256;

COMMODITIZATION OF ADDRESS SPACE

In almost all other cryptocurrencies, new users can simply create a new address for themselves at will. This creates a near-infinite address-space that can quickly bloat the blockchain without the commensurate user-base or demand to justify it. Both malicious and benign actors alike can exploit this ability for unbounded growth in the blockchain/DAG.

If the address space was instead made finite, it becomes a limited resource able to be commoditized. This is how the economics of Pascal Accounts (PASA) operates. In Pascal, accounts are limited in supply and minted into existence by the miner. The miner then distributes these accounts to users via a market mechanism. Pascal includes in-protocol PASA market that allows users to buy and sell accounts between themselves. Both private and public sale modes are supported. In version 5, atomic swapping of accounts (or coins) can be performed through the same

mechanism.

A commoditized address-space creates a natural space-saving mechanism since the underlying storage is not littered with unneeded or used keys. It also disincentivizes spammers, abuse or apathetic use of SafeBox. Most importantly, commoditization of the address space is a fundamental aspect of Pascal's Infinitely Scalable architecture.

SCALING ADDRESS-SPACE FOR GLOBAL ADOPTION

A common criticism of a commoditized address space is the inability to onboard "billions of users" instantly. Whilst it's true that Bitcoin could theoretically on-board the global population overnight in terms of addresses, those users would not be able to use those addresses due to the limited transaction throughput of the network. Only a tiny fraction of those users could use the network, if at all.

Conversely, Pascal's network does permit a global transaction throughput but paradoxically lacks addresses to cover the global population. Currently, there are over 1 million accounts and most of them are unused. Pascal adds one new account per minute to the SafeBox. At that rate, it would require 14,000 years to generate enough addresses to cover the current global population. Clearly, a solution is required to scale the address-space for real demand.

The solution to this issue is to increase the number of accounts being created by every block. PIP-0035 Block Policy[10] proposes an in-protocol governance model that permits existing users to govern how many accounts are created per block, ranging anywhere from 0 to 4095. This value can vary on block by block basis. As a result, during a burst of adoption the community can throttle the new account creation to a maximum of 1.1 million accounts per day. When the burst diminishes, it can be throttled back down in order to minimize storage.

This Block Policy proposal is slated Version 6 of the protocol. It should be noted that currently, accounts are in total abundance and virtually free. The Pascal developers (and community) are committed to ensuring genuine new users are able to acquire an account near-0 cost.

RELIABLE 0-CONFIRMATION TRANSACTIONS

In order to facilitate merchant adoption, Pascal implements a reliable 0-confirmation scheme suitable for everyday commerce. Typically, in a cryptocurrency like Bitcoin, a merchant accepting an unconfirmed (0-confirmation) transaction is at risk of a double-spend attack.

A double-spend attack occurs when a buyer pays for a good using a transaction but secretly double-spends those funds back to themself after merchant accepts payment. In order for this to work, the payment and double-spend transaction need to be published at near-simultaneous times but from geographically sparse locations. This way, the merchant node detects the "payment" first whilst a miner node detects the "double-spend" first. The merchant accepts the 0-confirmation but only learns later that it was double-spent when the miner mines the next block.

Pascal solves this issue by introducing Double-Spend Alerts (DSA). A DSA is network message alerting other nodes of a double-spend. It contains the account number committing the double-spend and a copy of the double-spend transaction. When a node detects a double-spend transaction it publishes a DSA to all its known peers.

Merchants who want to accept 0-confirmation transactions need only wait an additional 2-3 seconds and listen for a DSA involving the buyer's account. If no DSA was detected during that period, the merchant is almost guaranteed that the 0-confirmation payment transaction will be minted in a future block.

This works since Pascal uses a simplified account model instead of complicated UTXO-model (like Bitcoin). As a result, the detection of double-spends is a trivial matter for any node in the network. Processing a DSA is computationally inexpensive and merchants can filter irrelevant DSAs by using the account number.

The only alternative left to the attacker is to collude with a miner to secretly mine the double-spend transaction. Since this class of attack is "industrial scale", it's beyond the domain of the ordinary thief but well within the risk-tolerance of everyday merchants. As a result, merchants who use DSA's can reliably accept 0-confirmations for every day commerce. However, as always, transactions involving large amounts should always wait a commensurate number of block confirmations.

ZERO-FEE TRANSACTIONS

Another unique feature of Pascal 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 simply enforce that a public key can have a maximum of one zero-fee transaction in a block (or memory pool) at any time. This simple approach addresses 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 minimal fee is required (currently set at 0.0001 PASC). Zero-fee transactions are achieved on the merits of Pascal's ultra-high throughput in which the fee bidding competition for transaction priority (typical in other UTXO cryptocurrencies) is moot. The fee pressure only starts rising as current throughput approaches maximum throughput, similar how Bitcoin's fee market works. However, since Pascal's current architecture achieves VISA-scale throughput (on average) and the roadmap intends to deliver orders of magnitude higher throughput, transaction fees are always expected to remain zero for infrequent usage.

Zero-fee transactions apply only to "vanilla" transactions, namely those that are not used for Layer-2 purposes or batch transactions. Such transactions will involve negligible fees.

ACCOUNT NUMBERS, NAMES AND TYPES

Pascal 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. However, unlike most cryptocurrencies, Pascal accounts are human readable and writable numbers (e.g. 1234-56) and not complicated strings that must be copy-pasted or scanned.

One of the key features of Pascal is that accounts can have unique names that are publicly visible, much in the same way as the Domain Name System (DNS). This allows users to send and receive funds to and from their email addresses, social media monikers, business and brand name, etc. Payments still refer to accounts via their numbers, but the name is used by the sender to look up the recipient in a trust-less, permission-less manner much like how an IP address is looked up from Domain Name.

Also, an Account has a Type field that is a number that ranges from 0 to 65535 and a Data field containing 32 bytes. These fields serve a fundamental purpose in Layer-2 use-cases. For example, Accounts with Type "12356" could be agreed (via social convention) to refer to "PascalChat Application". The Account Name can be used as the "Chat Room Name". Transactions to the account can be interpreted as Chat Messages and their Payloads containing the "Chat Message". A more sophisticated application could instead remove chat messages completely off-chain by simply using the Account Data as "state hash" of the Layer-2 database containing the chat messages.

This allows an account to serve as a consensus mechanism for Layer-2 applications. Such consensus mechanisms are discussed later.

CRYPTOGRAPHIC INTEGRITY

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 each Account Segment created by that block. 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 via the target difficulties in the block headers; the total work used to evolve that state can thus be calculated by examination of the SafeBox itself without any blocks. Nodes will only adopt the SafeBox with the most work.

RE-ORG CHECKPOINTING

Pascal nodes are only required to keep the last 100 blocks. This means the SafeBox state before the 100 blocks cannot be validated at a transactional level. It is assumed that since the SafeBox is the "most-work" one, the state of accounts before the last 100 blocks are valid. Only the operations (transactions) within the last 100 blocks are necessarily validated.

This introduces a "State-Attack" scenario where a malicious actor secretly mines a corrupted SafeBox with majority hashpower for an extended period of time. After 100 blocks, the malicious miner publishes the corrupted SafeBox to the network as the true and correct SafeBox. Since the corrupted SafeBox has more aggregated proof-of-work than the current honest SafeBox, and all the transactions within the last 100 blocks of the corrupted SafeBox are valid, the honest network is fooled into accepting the corrupted SafeBox.

To resolve this issue, Pascal imposes a 100 block re-org limit on the network. This means a node will not accept a new SafeBox unless it forked from the current SafeBox within the last 100 blocks. In this manner, a State Attack scenario is thwarted since the malicious miner will fail to propagate their corrupted SafeBox to the honest network.

Similar approaches have been employed by other cryptocurrencies successfully. For example, Bitcoin Cash employs a 10 block re-org limit. In the view of many, a re-org of more than 10 blocks is indicative of other serious issues. As a result, whilst this approach may appear controversial in the eyes of some in is inconsequential to practical security in the eyes of others. However, the Pascal developers are also researching alternative approaches to resolving State Attack using fraud-proofs involving cryptographically secure account histories. If a solution emerges through this approach, re-org checkpointing can be safely replaced at a later date.

PRIVATE TRANSACTIONS

Due to Pascal'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, Pascal is also committed to providing equally anonymizing features for those use cases where privacy is important. As a result, Pascal has a strong privacy roadmap that has already been partially rolled out.

In Pascal version 1, users could transfer PASC privately using a PASA-exchanging approach. Instead of sending PASC 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 the PASC payment.

Pascal's version 3 added in-protocol PASA and PASC mixing capability. This capability will allow future roll-out of in-protocol tumbling workflow similar to CoinShuffle[12] / CashFusion[13] pioneered by Bitcoin Cash. Through this approach, users who elect to make a private transaction would only pay a marginal fee and, under the hood, the wallet would obfuscate their transaction beyond any ability to reconstruct it. It does this by initiating a network-wide workflow between many nodes who collectively co-create a large transaction that sends PASC and exchanges PASA between a variety of accounts and keys. Since the resulting transaction involves many participants, none of whom know any information about each other, the original sender, recipient and amount are irreversibly obfuscated. By chaining many of these transactions together in rapid succession, the obfuscation-based anonymity increases by orders of magnitude for little extra cost.

In addition, nodes will be able to earn fees by offering their latent PASC and PASA for use in such mixing transactions. This allows them to earn residual income whilst providing the network a rich set of PASC/PASA to participate in the anoymity. This would result in a fast, cheap and seamless anonymity for Pascal users. The roadmap also includes R&D proposals for Layer-1 zero-knowledge proofs and Layer-2 dApps to achieve other avenues for anonymity.

ATOMIC SWAPS

Atomic swaps are a smart-contract technology that enables the exchange of one cryptocurrency for another without using intermediaries (such as 3rd party exchanges). Atomic swaps occur as smart-contract events within the respective blockchains in a cryptographically secure manner. Atomic swaps are cornerstone of DEX capability.

Pascal version 5 introduces atomic swap capability [15] via the in-protocol account exchanging mechanism. An Account can be time-locked and hash-locked to a counterparty granting that counterparty the option to take ownership of that account (or the coins within) by simply unlocking the hash-lock within the time-lock period.

COINROT AND ACCOUNT RECOVERY

Coin-rot is the phenomenon of coins being permanently lost due to lost/corrupted keys and/or natural owner death and has become a noteworthy issue afflicting cryptocurrencies. For example, it is estimated that approximate 20% of BTC have "rotted." Pascal solves this issue by allowing an inactive account to be repossessed by a miner after a 4-year period. An account is considered inactive if no state change has occurred within 4 years.

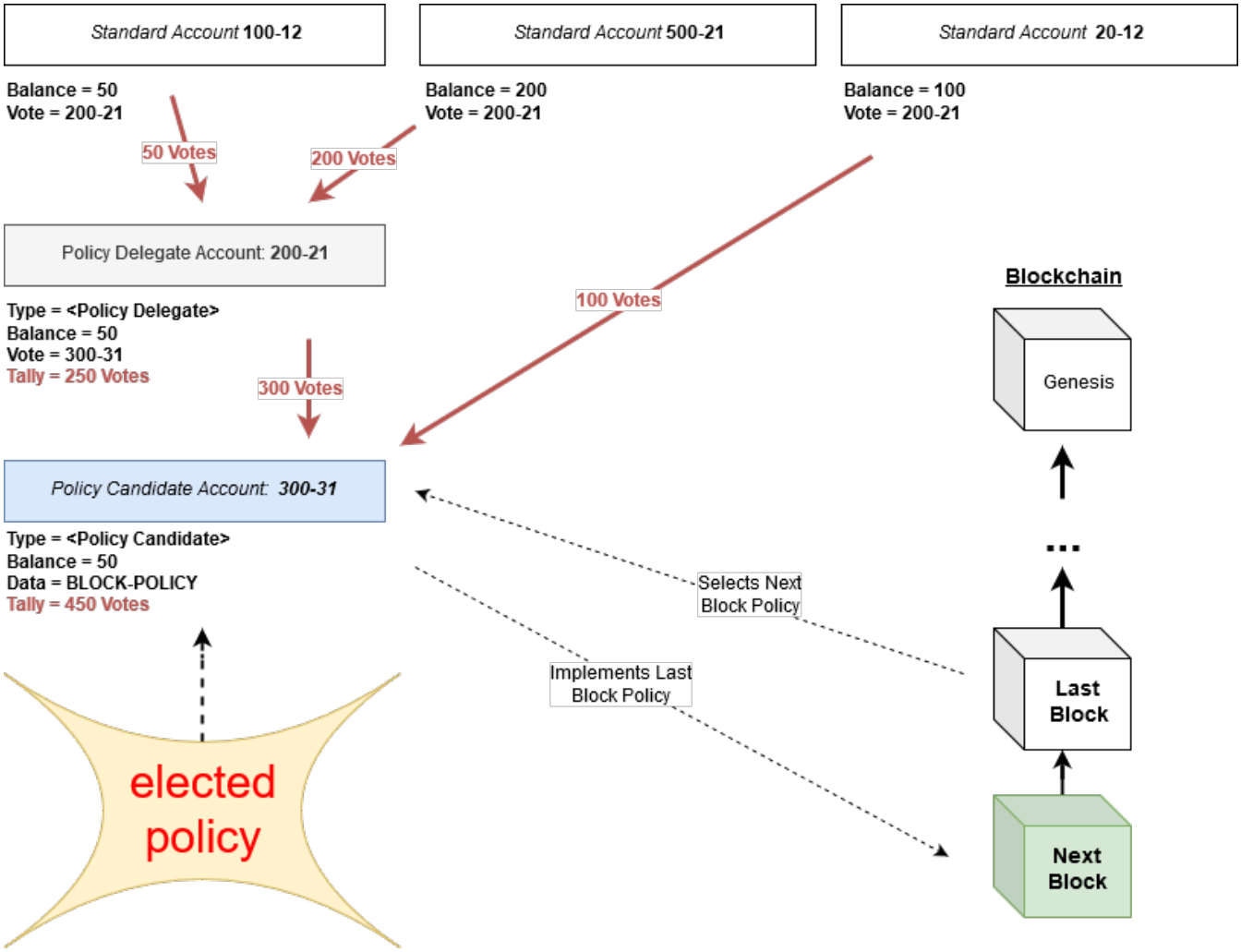
BLOCK-POLICY & IN-PROTOCOL GOVERNANCE

A simple and efficient layer-1 governance model is proposed for version 6 called "Block Policy"[10]. Whilst only a relatively minor technical change, Block Policy offers a significant innovation that solves many of the issues afflicting decentralized consensus systems today.

Block Policy proposes to:

- Re-implement existing technical and economic protocol constants as dynamic variables.
- Allow accounts to vote on what these variables should be using a SafeBox signaling mechanism.
- Allow these variables to change on a block-by-block basis, enabling real-time governance.
- No longer award new accounts to miners, instead issue them without ownership and list them for public sale at a floating price.

This proposal introduces in-protocol real-time governance in layer-1 and establishes a cryptographically secure voting mechanism that can be audited by anyone, anytime, permission-free and instantly. By moving economic constants from the source-code and into the Block Policy, the layer-1 protocol is now fundamentally enhanced with "external-world feedback". This has far-reaching consequences at every level of the project including security, usability, adoption, governance and market price.



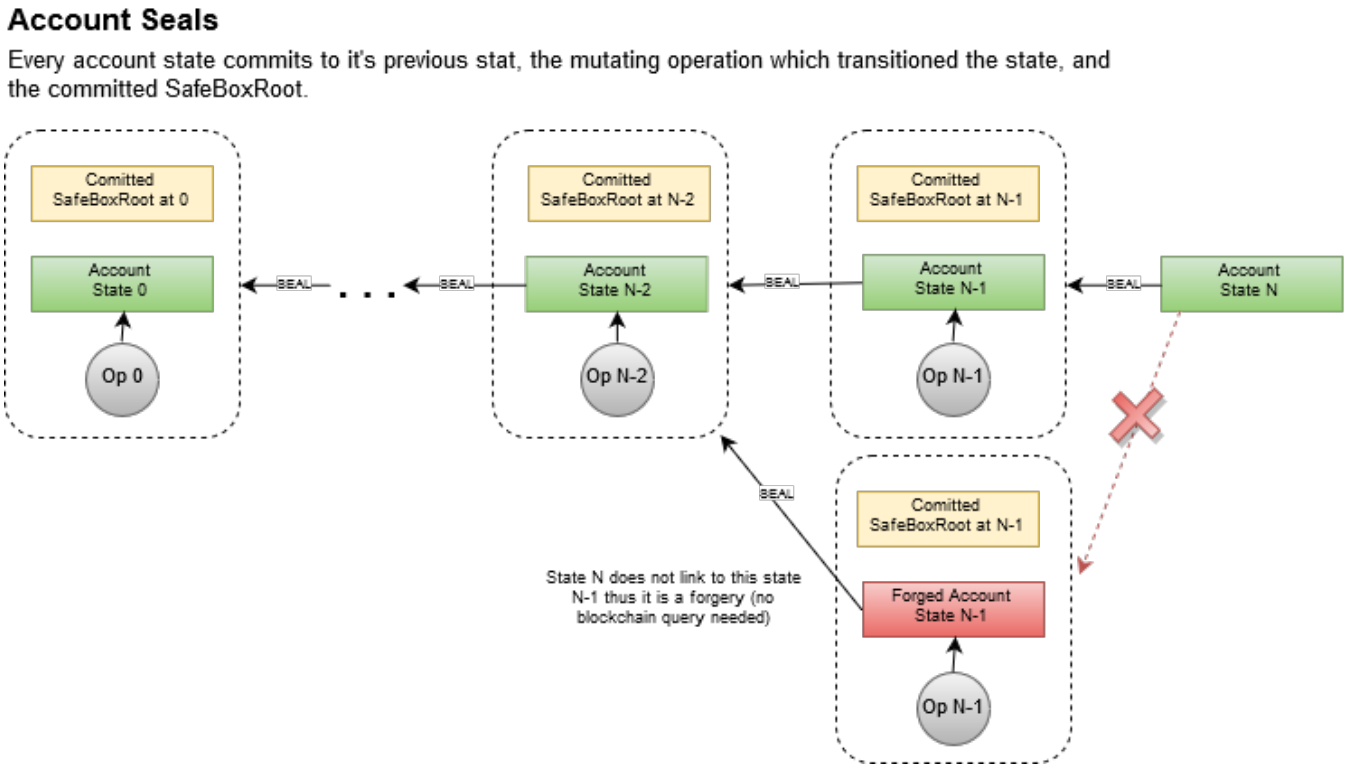
CRYPTOGRAPHICALLY SECURE ACCOUNT HISTORIES

Transactions in Pascal mutate account states in a variety of ways (e.g. balance, name, key, data, etc.) and follow strict consensus rules enforced by all nodes on the network. Every time an account state is changed, its prior state is lost from the SafeBox. Since the Pascal network is only required to maintain the last 100 blocks, an account history is essentially lost from the network after 100 blocks.

In order to know more than 100 blocks of history, the user must track these operations manually, or request them from an archival node. A dilemma arises here in that without the full blockchain, there is no way for a user to determine if an account history is forged, corrupt, invalid or even valid. The user is left with a trust-model. In a trust-free scenario, such a history is essentially useless since it is not intrinsically verifiable.

Pascal resolves this issue by using "Account Seals"[7]. Every account contains a 20 byte field called the "Seal" which is a hash commitment to its previous state and the transaction which mutated that state. This results in a cryptographically verifiable chain of account states.

Since each state links to the prior state, and that prior state links to its prior state, an Account Seal commits to the entire preceding history of that account. Therefore, a user possessing an account history can verify that history by re-calculating the seals in chronological order ensuring that the final seal matches the current SafeBox seal.



LAYER-2 ARCHITECTURE

Pascal Layer-1 was designed as a scalable financial layer that transfers value between accounts. The major use-case here is payments. Pascal's technology does payments very well, arguably better than most other cryptocurrencies since it is not burdened with any scripting complexity.

However, the SafeBox design permits new and compelling use-cases that essentially re-purpose Pascal as a consensus and financial layer for external Layer-2 protocols. In these use-cases, Pascal serves as a Layer-1 protocol that supports a Layer-2 protocol in much the same way TCP/IP is a base protocol for HTTP. These Layer-2 protocols, applications and networks are collectively referred to as "dApps".

dApps operate separately from Pascal, use their own protocols, their own network connections and their own storage. They can integrate into Pascal in a variety of ways. Whilst the full technology-stack will not be disclosed in this white-paper, some discussion will be provided.

DATA OPERATIONS

Pascal provides an operation called OP_DATA that allow an account to send a data packet to another account. These packets can be public or encrypted. Encryption modes include ECIES encryption using sender or recipient key, or AES using a shared secret. These data packets are 255 bytes in length and include a 16 byte GUID key and 16 bit Sequence field that can be used to group packets into a larger logical data-stream. This approach provides a clear enveloping capability for Layer-2 protocols much in the same way that TCP/IP envelopes HTTP in networking.

DECENTRALIZED CONSENSUS LEDGER

Data Operations combined with cryptographically secure account histories can be used to implement decentralized consensus ledgers for a wide-variety of use-cases. Under this interpretation, an account history becomes a ledger of chronologically ordered statements made by Layer-2 nodes which cannot be forged, altered or tampered with by those nodes. These data messages embed the Layer-2 messages conforming to a Layer-2 protocol. Should any dispute arise between those Layer-2 nodes, this Layer-1 account history serves as the immutable audit log and authority of all the Layer-2 transactions. Layer-2 consensus emerges merely by examination of the Layer-1 account history alone. No trust is required between Layer-2 nodes whatsoever.

Since an account history is cryptographically secure and intrinsically verifiable, Layer-2 nodes can validate their consensus ledgers without the Layer-1 blockchain or SafeBox. They only need to ensure the latest account state matches the account state in most proof-of-work SafeBox. This can be achieved by requesting a merkle-proof of the parent Account Segment from the Layer-1 network, a virtually instant operation.

PROOF-OF-STAKE OVERLAY NETWORK

Currently Layer-2 networks can utilize Layer-1 Pascal as a consensus-only layer, but not as a financial layer. In order to complete the Layer-2 integration, Pascal requires a mechanism to enable Layer-2 networks to authorize funds

from Layer-1 Pascal accounts in a manner that Layer-1 miners are not required to participate in the Layer-2 business logic. The technology that permits Layer-2 networks to achieve this has been fully designed. The details of this technology remain undisclosed at this point in time until prototyping has been completed and verified.

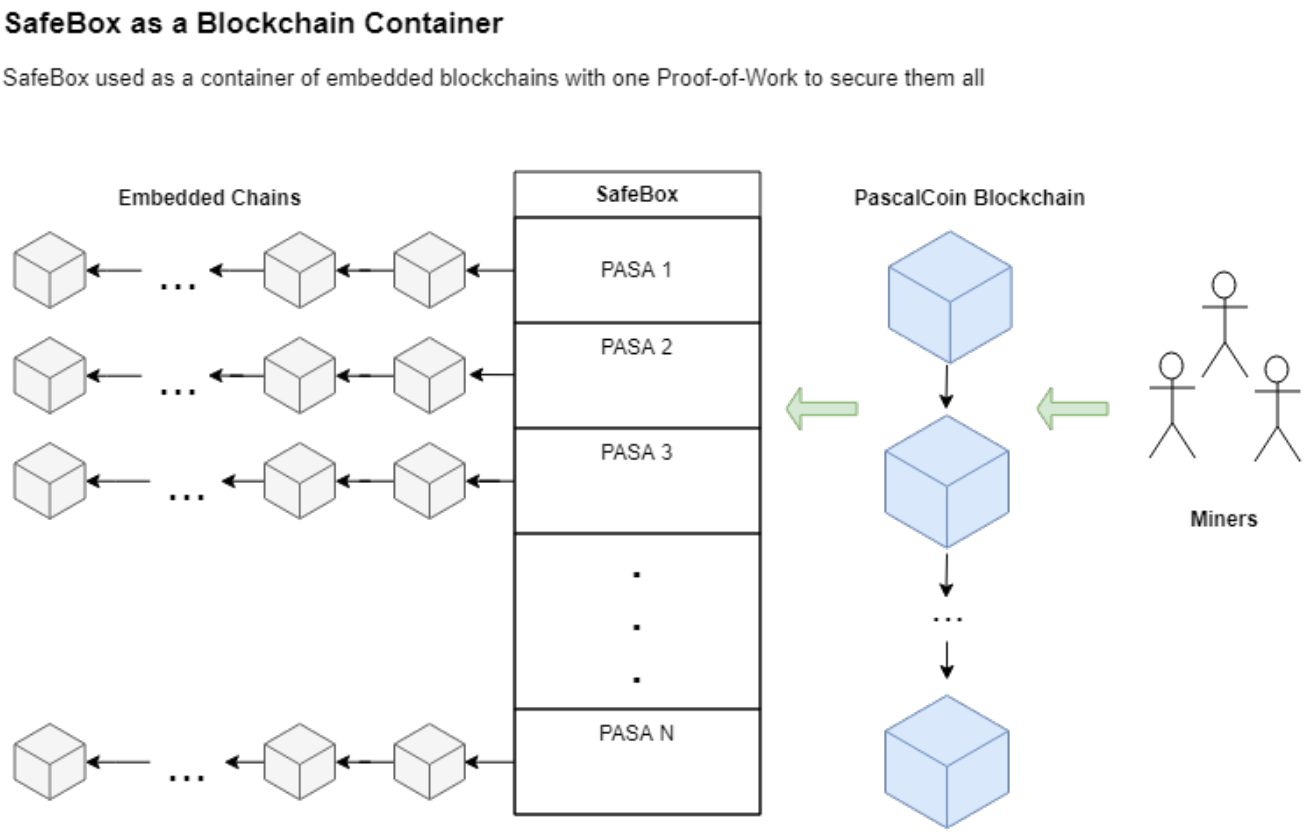
MONETIZED APIs

Due to Pascal's account-based model, a new form of dApp is enabled called Monetized API. A Monetized API is a reconceptualization of an account as a message-queue. Data-operations into the message-queue serve as "requests" and similarly, data-operation from the message-queue serve as "responses". Since requests and responses contain data and monetary value, a monetized API system is created.

Monetized APIs are already in deployment such as GetPasa.com, which receives requests containing a public key and responds by sending an account to that key. Similarly, an entire class of "smart-agent" applications are currently enabled by Pascal's SafeBox architecture.

EMBEDDED CHAINS

The decentralized consensus ledger scheme described above can also be used to maintain a set of block-headers of a side-chain, referred to as an “embedded-chain”. The contents of the blocks from embedded-chains are not necessarily required to be included, only the headers. Under this interpretation, the SafeBox becomes a “Blockchain Container” capable of securing a myriad of blockchains (1 per account) where only one Layer-1 proof-of-work secures them all.



ACKNOWLEDGEMENTS

The authors appreciate the suggestions and editing of this whitepaper from Tyler Swob.

REFERENCES

[1] Schoenfeld, H (2018). PIP-0010 - 50% Inflation Reduction. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0010.md*

[2] Schoenfeld, H (2018). PIP-0011 - 20% Developer Reward. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0011.md*

[3] Schoenfeld, H (2018). PIP-0017 - MULTI-OP: atomic batch operations. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0017.md*

[4] Schoenfeld, H (2017). PIP-0009 - RandomHash: GPU & ASIC Resistant Hash Algorithm. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0009.md*

[5] Schoenfeld, H (2018). PIP-0016 - DATA-OP: In-Protocol Data Exchange (Layer-2 support). URL *https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0016.md*

[6] Schoenfeld, H (2018). PIP-0015 - Fast Block Propagation. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0015.md*

[7] Schoenfeld, H (2019). PIP-0029 - Account Seals: Cryptographically Secure Account Histories. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0029.md*

[8] Schoenfeld, H (2019). PIP-0030 - SafeBoxRoot: Deletable SafeBox and Light-Nodes. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0030.md*

[9] Schoenfeld, H (2019). PIP-0027 - E-PASA: Infinite Address-Space (via Layer-2). *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0027.md*

[10] Schoenfeld, H (2019). PIP-0035 - Block Policy: Layer-1 Governance. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0035.md*

[11] Schoenfeld, H (2019). PIP-0036 - RandomHash2: Enhanced GPU & ASIC Resistant Hash Algorithm. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0035.md*

[12] Tim Ruffing, Pedro Moreno-Sanchez, Aniket Kate. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. *URL https://crypsys.mmci.uni-saarland.de/projects/CoinShuffle/coinshuffle.pdf*

[13] Jonald Fyookball, Mark B. Lundeberg. CashFusion. *URL https://github.com/cashshuffle/spec/blob/master/CASHFUSION.md*

[14] Schoenfeld, H (2017). PIP-0003 - Infinite Scaling via Deletable Blockchain. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0003.md*

[15] Schoenfeld, H (2019). PIP-0032A - Atomic Swaps via Hash-Locked Accounts. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PIP/PIP-0032A.md*

[16] Molina, A (2016). Pascal Coin: Crypto currency without need of historical operations. *URL https://github.com/PascalCoin/PascalCoin/blob/master/PascalCoin%20White%20Paper%20-%20EN.pdf*