Whitepapers

Want a more in-depth understanding of Pascal?

Below you can download the Whitepaper which goes into greater detail on the inner workings of Pascal, including the groundbreaking SafeBox technology.

Whitepaper V5 (English)
Version 5.0, SEPT 2019

Whitepaper V5 (Chinese)
Version 5.0, SEPT 2019

Extended Information

RandomHash - GPU & ASIC Resistant Hash Algorithm

Pascal adopts a low-memory, GPU- and ASIC-resistant hash algorithm called Random Hash - the first of its kind - that fully preserves mining decentralization. Random Hash is an innovative, "high-level cryptographic hash" algorithm that combines other well-known hash primitives in a highly serial manner. In addition to RandomHash's serial nature, it is branch-heavy and recursive which makes Pascal optimal for CPU-only mining. RandomHash is designed to ensure Pascal remains a globally decentralized network that runs well on low-end hardware.

Technical specifications for Random Hash can be found in its dedicated whitepaper below.

RandomHash Whitepaper

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 their architectures as it imposes a maximum viable usability age to their networks.

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 occasionally, 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.

Instant Zero-Fee Transactions

Reliable 0-Confirmation Transaction

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-spent those funds back to themself without the merchant knowing. In order for this to work, the payment and double-spend transaction need to be published at near-simultaneous times but from geographically sparse regions. This way, the merchant node detects the payment transaction 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 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 it's 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 disregard DSAs not relevant to them 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. 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 extremely 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 improvements to this throughput, transaction fees are always likely to remain zero for infrequent usage.

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 & 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-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, Account's have a Type field that is a number that ranges from 0 to 65535 and a Data field of 32 bytes. These fields serve a fundamental purpose in Layer-2 consensus protocols. For example, Accounts with Type "12356" could be agreed (by social convention) to refer to "Chat Application". The Account Name can be used as the "Chat Room Name". Transactions to the account can be interpreted as Chat Messages, and the 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.

Layer-2 Applications

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

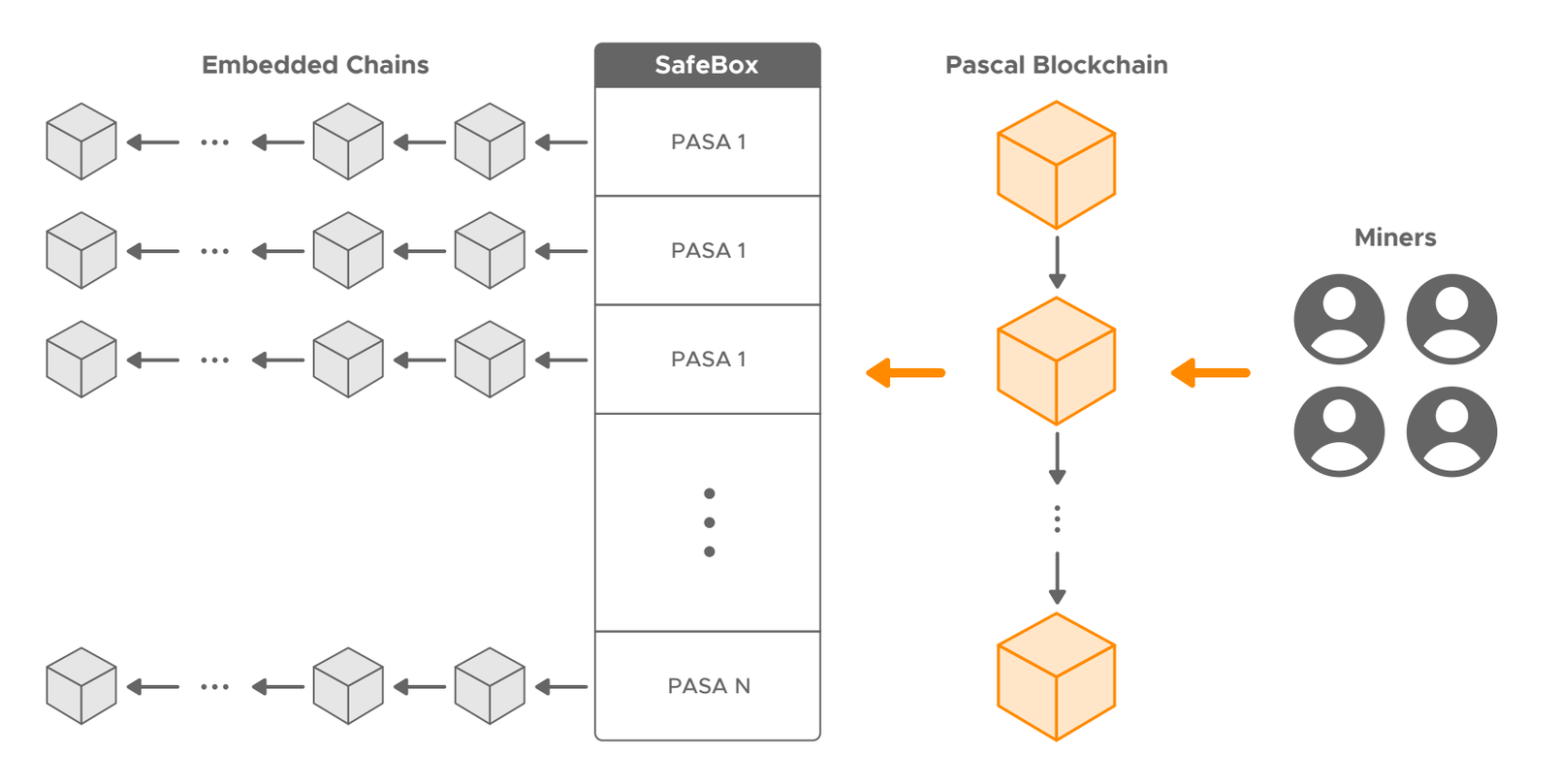
Data Operations combined with cryptographically secure account histories can be used to implement decentralized consensus ledgers for a wide-variety of use-cases. Under this interpretation, an account history becomes a ledger of chronologically ordered statements made by Layer-2 nodes which cannot be 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.

Poof-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.

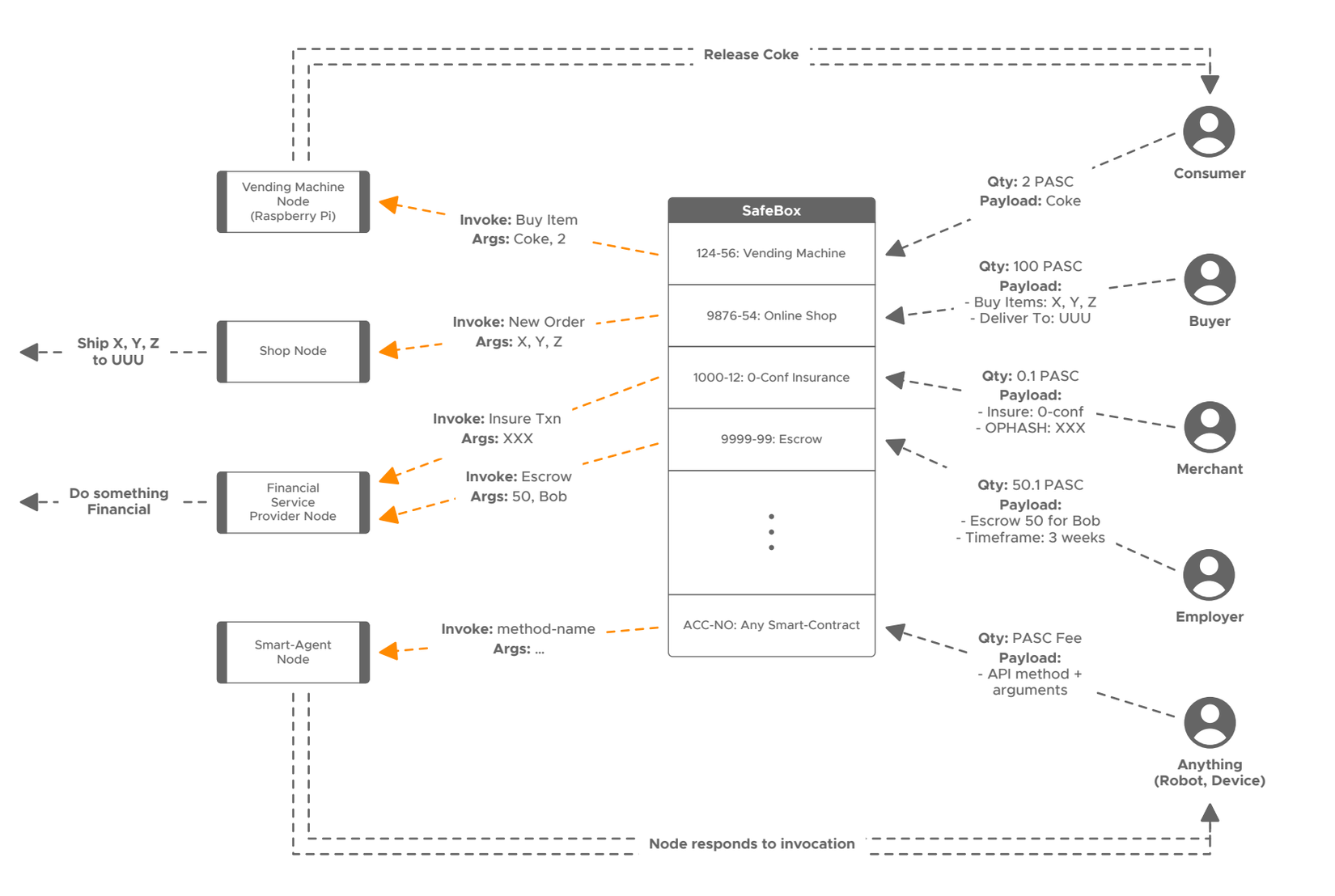
Embedded Chains

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.



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.



Governance & Treasury

Pascal was launched 100% fairly without any pre-mine, ICO or investment rounds. This led to significant developer and community growth. However, due to the reality of the cryptocurrency sector, PIP-0011 was proposed and voted upon by the community. Based on PIP-0011, the Pascal Foundation was established and receives 20% of the mining reward. This reward is then made available to the community in the form of a Decentralised Autonomous Organisation (DAO). Besides the funding, the Pascal Foundation also ensures a strong and fully decentralised governance system for all aspects of the Pascal project.

Anyone can make proposals for funding but only HODL'ers of PASC can vote on these proposals and trigger the funding release (or abort). Voting is real-time and automated via a GOVBOT Pascal smart contract. The funding is intended to further every aspect of Pascal including core development, layer-2 development, exchange listings, social promotions, meetup sponsorships, affiliate sponsorships, etc.

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 a PASC transaction.

Pascal's version 3 added an in-protocol PASA and PASC tumbling capability. This protocol capability will allow future roll-out of anonymity approaches similar to CashShuffle[12] and 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 nodes who all end up co-creating 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 which know any information about each other, and each are sending PASC and PASA to unknown participants, the original sender, recipient and amount are irreversibly obfuscated. By chaining many of those mixing transactions together in rapid succession, the obfuscation 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 such mixing transactions whilst simultaneously providing a rich set of decentralized PASC/PASA to participate in the tumbling. 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.

Data Transfer

Pascal'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 (OP_DATA) to the receiver. The 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 Pascal'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 decentralised document management solutions for both private and public data.

Original Codebase

Pascal was designed and written from scratch in the Pascal programming language without copying a line of code from any other project. The development team, led by Albert Molina, are mature, exceptional developers with proven track records of delivering high-quality software. Pascal has withstood 2 years of open attacks, and survived. The code works and is improving every day.

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 Pascal programming is used for Pascal's core development, DApps and any external implementations based on Pascal can be written in any language.