Bootstrapping Trust in Distributed Systems with Blockchains

Muneeb Ali[⋆†], Jude Nelson[⋆†], Ryan Shea[†], Michael J. Freedman[⋆]
[⋆]*Princeton University,* [†]*Blockstack Labs*

1 Introduction

Cryptocurrency blockchains and their respective P2P networks are useful beyond exchanging money. They provide cryptographically auditable, append-only global logs that have a high computational-cost barrier for tampering with data written to them. Blockchains have no central points of trust or failure: they minimize the degree to which users/nodes need to trust a single party, like a DNS root server or a root certificate authority.

Blockchain networks have attracted a lot of interest from enthusiasts, engineers, and investors. In fact, $1.1 billion has been invested in blockchain startups over the past few years [5]. With the rapid capital infusion, infrastructure for blockchains is getting quickly deployed and they are emerging as publicly available common infrastructure for building services and applications.

Blockchains are already being used to build new, *decentralized* versions of DNS[1] and public-key infrastructure (PKI)[2] that have no central points of trust. Such blockchain-based naming and PKI services can be used as a general-purpose "trust layer" for other distributed systems and applications. For example, an IoT node can be registered on a blockchain with a unique name and controlled by a cryptographic keypair binding, stored on the blockchain, with that name.

Blockchain-based naming systems securely bind *names*, which can be human-readable, to arbitrary *values*. The blockchain gives consensus on the global state of the naming system and provides an append-only global log for state changes. Writes to name-value pairs can only be announced in new blocks, as appends to the global log. The global log is fully replicated (all nodes on the network see the same state), but organizationally decentralized (no central party controls the log).

The decentralized nature of blockchain-based naming introduces meaningful security benefits, but certain aspects of contemporary blockchains present technical limitations. Individual blockchain records are typically on the order of kilobytes [10] and cannot hold much data. The latency of creating and updating records is capped by the blockchain's write propagation and leader election protocol, and it is typically on the order of 10-40 minutes [4]. The total number of new operations in each round are limited by the average bandwidth of the network's nodes; for Bitcoin, the current average is ∼1500 new operations per new round [5]. Further, new nodes need to independently audit the global log from its beginning: as the system makes forward progress, the time to bootstrap new nodes increases linearly.

We believe that in spite of these scalability and performance challenges, blockchains provide important infrastructure for bootstrapping trust in distributed systems and building new decentralized services. The cost of tampering with blockchains grows with their adoption: today, it would require hundreds of millions of dollars to attack a large blockchain like Bitcoin [5].

These benefits motivated us to use blockchains to build a new global naming and storage system, called Blockstack [3]. Our system enables users to register unique, human-readable usernames and associate public-keys, like PGP, along with additional data to these usernames. In this article, we present an overview of the design of Blockstack and discuss how it can be used as a general-purpose "trust layer" for building other applications and services. Unlike previous blockchain-based systems, Blockstack *separates its control and data plane considerations*: it keeps only minimal metadata (namely, data hashes and state transitions) in the blockchain and uses external datastores for actual bulk storage. Blockstack enables fast bootstrapping of new nodes by using checkpointing and *skip lists* to limit the set of blocks that a new node must audit to get started.

Modifying production blockchains like Bitcoin (and introducing new functionality for which it was not designed) is quite difficult, particularly that the system still

[1] `http://namecoin.info`
[2] `http://onename.com`

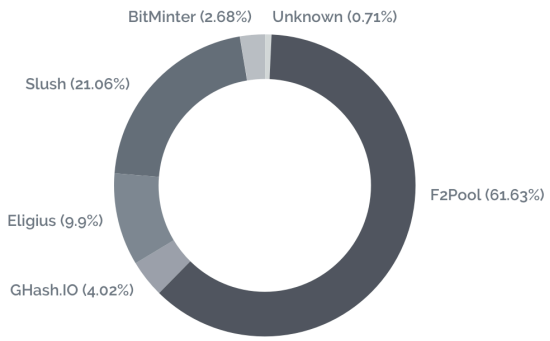

Figure 1: Mining distribution for Namecoin (Aug'15).

needs to reach "consensus." With Blockstack, we extend the single state machine model of blockchains to allow for arbitrary state machines without requiring consensus breaking changes in the underlying blockchain. This design was non-intuitive before our work; indeed, the standard approach for the past three years was to fork the main Bitcoin blockchain to add new and different functionality. We have released Blockstack as open source.[3]

2 Motivation and Background

We next describe the motivation for building naming systems that have no central point of trust and provide the relevant background on blockchains. We use the term *naming system* to mean (a) names are **human-readable** and can be picked by humans, (b) name-value pairs have **a strong sense of ownership**—that is, they can be owned by cryptographic keypairs, and (c) there is **no central trusted party** or point of failure. Building a naming system with these three properties was considered impossible according to Zooko's Triangle[4] and most traditional naming systems provide two out of these three properties [8]. Namecoin used a blockchain-based approach to provide the first naming system that offered all three properties: human-readability, strong ownership, and decentralization.

2.1 Background on Blockchains

Blockchains provide a global append-only log that is publicly writeable. Writes to the global log, called *transactions*, are organized as *blocks* and each block packages multiple transactions into a single atomic write. Writing to the global log requires a payment in the form of a *transaction fee*. Nodes participating in a blockchain network follow a leader election protocol for deciding which node gets to write the next block and collect the respective transaction fees. Only one node gets to write a block in each leader election round. Not all nodes in the network participate in leader election. Those actively competing to become the leader of the next round are called *miners*. At the start of each round, all miners start working on a new computation problem, derived from the last block, and the miner that is the first to solve the problem gets to write the next block. In Bitcoin, the difficulty of these computation problems is automatically adjusted by the protocol so that one new block is produced roughly every 10 minutes. See [4] for further details on how blockchains work and how they reach consensus.

2.2 Naming System on a Blockchain

The first blockchain to implement a naming system was Namecoin. It is one of the first forks of Bitcoin and is the oldest blockchain other than Bitcoin that is still operational. The main motivation for starting Namecoin was to create an alternate DNS-like system that replaces DNS root servers with a blockchain for mapping domain names to DNS records [8]. Given that blockchains don't have central points of trust, a blockchain-based DNS is much harder to censor and registered names cannot be seized from owners without getting access to their respective private keys [8]. Altering name registrations stored in a blockchain requires prohibitively high computing resources because re-writing blockchain data requires *proof-of-work* [2]. Before our work, it was common practice to start new blockchains (by forking them from Bitcoin) to introduce new functionality and make modifications required by the respective service/application, which is the precise approach taken by Namecoin.

Just like DNS, there is a cost associated with registering a new name. The name registration fee discourages people from registering a lot of names that they don't actually intend to use. In Namecoin, the recipient of registration fees is a "black hole" cryptographic address from which money cannot be retrieved [8]. Namecoin defines a pricing function for how the cost of name registrations changes over time. Namecoin supports multiple namespaces (like TLDs in DNS), and the same rules for pricing and name expiration apply to all namespaces. By convention, the $d/$ namespace is used for domain names.

In Namecoin, name registration uses a two-phase commit method where a user first **pre-orders** a name hash and then **registers** the name-value pair by revealing the actual *name* and the associated *value*. This is done to avoid front-running of unconfirmed name registrations [8]. Name registrations expire after a fixed amount of time, measured in new blocks written (currently 36,000 blocks, which translates to roughly 8 months), and cannot be purchased for longer periods. Namecoin

[3] `http://github.com/blockstack`
[4] `http://dankaminsky.com/2011/01/13/spelunk-tri`

also supports updating the value associated with a name, as well as ownership transfers.

Our experience with the Namecoin blockchain shows that starting new, smaller blockchains leads to security problems (like reduced computational power needed to attack the network) and should be avoided when possible. We detailed our findings in a recent USENIX ATC paper [3]. **Importantly, we discovered a critical security problem where a single miner consistently had more than 51% of the total compute power on the Namecoin network** (see Figure 1 for the Namecoin mining distribution for the month of August 2015 and see [4] for details on the 51% attack and compute power of miners). A 51% attack is one of the most serious attacks on a blockchain and impacts its security and decentralization properties. Other than Namecoin, blockchains like Ethereum [1] and BitShares[5] also have support for human-readable names (tied to their respective blockchains).

3 Design of Blockstack

Blockstack is designed to implement a naming system with human-readable names in a layer above the blockchain. In this section, we describe how Blockstack uses the underlying blockchain, and present how it copes with technical limitations of contemporary blockchains.

3.1 Challenges

Building systems with blockchains presents challenges:

• **Limits on Data Storage:** Individual blockchain records are typically on the order of kilobytes [10] and cannot hold much data. Moreover, the blockchain's log structure implies that *all* state changes are recorded in the blockchain. All nodes participating in the network need to maintain a full copy of the blockchain, limiting the total size of blockchains to what current commodity hardware can support. As of May 2016, Bitcoin nodes need to dedicate 69GB total disk space to blockchain data for staying synchronized with the network.

• **Slow Writes:** The transaction processing rate is capped by the blockchain's write propagation and leader election protocol, and it is pegged to the rate at which new blocks are announced by leader nodes (*miners*). New transactions can take several minutes to a few hours to be accepted.

• **Limited Bandwidth:** The total number of transactions per block is limited by the *block size* of blockchains. To maintain fairness and to give all nodes a chance to become leader in the next round, all nodes should receive a newly announced block at roughly the same time. Therefore, the *block size* is typically limited by average uplink bandwidth of nodes. For Bitcoin, the current block size is 1MB (∼1000 transactions).

• **Endless Ledger:** The integrity of blockchains depends on the ability for anyone to audit them back to their first block. As the system makes forward progress and issues new blocks, the cost of an audit grows linearly with time. Thus, booting up new nodes becomes progressively more time consuming. We call this the *endless ledger problem*. As of May 2016, Bitcoin's blockchain had ∼413,000 blocks and new nodes take 1-3 days to download the blockchain from Bitcoin peers, verify it, and boot up.

3.2 Architecture Overview

Blockstack maintains a naming system as a separate logical layer on top of the underlying blockchain on which it operates. Blockstack uses the underlying blockchain to achieve consensus on the state of this naming system and bind names to data records. Specifically, it uses the underlying blockchain as a communication channel for announcing state changes, as any changes to the state of name-value pairs can only be announced in new blockchain blocks. Relying on the consensus protocol of the underlying blockchain, Blockstack can provide a total ordering for all operations supported by the naming system, like name registrations, updates, and transfers.

Separation of the Control and Data Plane: Blockstack decouples the security of name registration and name ownership from the availability of data associated with names by separating the control and data planes.

The control plane defines the protocol for registering human-readable *names*, creating $(name, hash)$ bindings, and mapping name ownership to cryptographic keypairs. The control plane consists of a blockchain and a logically separate layer on top, called a *virtualchain*.

The data plane is responsible for data storage and availability. It consists of (a) zone files for discovering data by hash or URL, and (b) external storage systems for storing data (such as S3, IPFS [6], and Syndicate [7]). Data values are signed by the public keys of the respective name owners. Clients read data values from the data plane and verify their authenticity by checking that either the data's hash is in the zone file, or the data includes a signature with the name owner's public key.

We believe this separation is a significant improvement over Namecoin, which implements both the control and the data plane at the blockchain level. Our design not only significantly increases the data storage capacity of the system, but also allows each layer to evolve and improve independently of the other.

Agnostic of the Underlying Blockchain: The de-

[5] `http://bitshares.org`

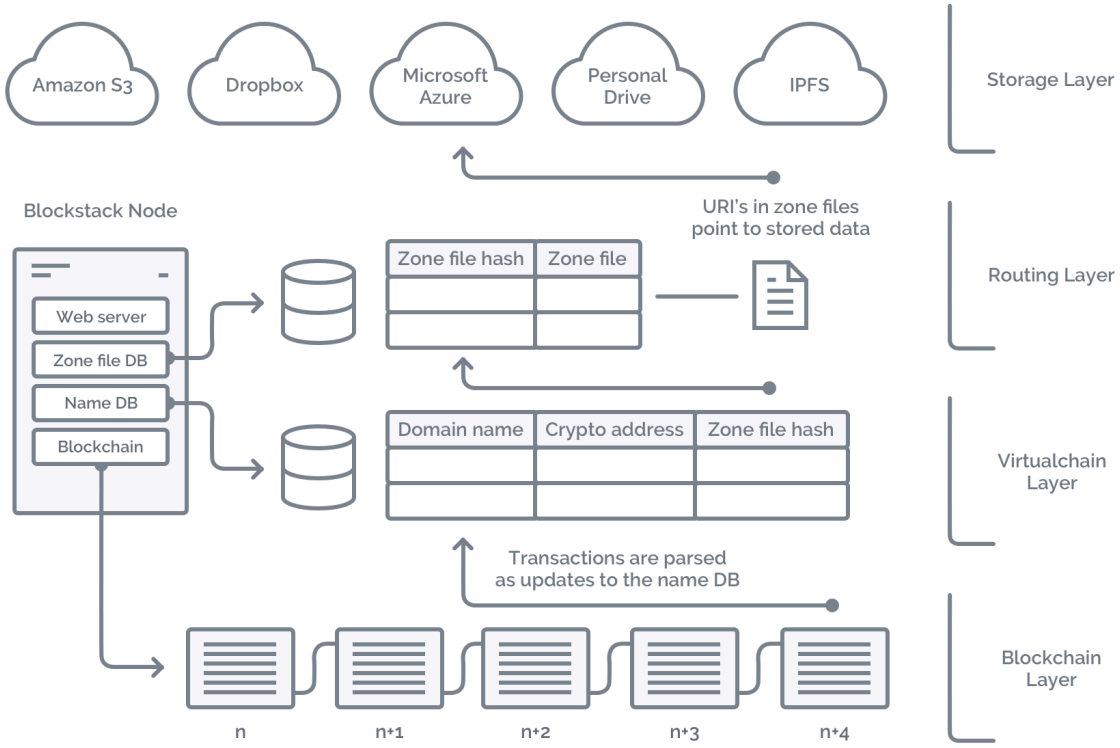

Figure 2: *Overview of Blockstack's architecture. Blockchain records give (name, hash) mappings. Hashes are looked up in routing layer to discover routes to data. Data, signed by name owner's public-key, is stored in cloud storage.*

sign of Blockstack does not put any limitations on which blockchain can be used with it. Any blockchain can be used, but the security and reliability properties are directly dependent on the underlying blockchain. We believe that the ability to *migrate* from one blockchain to another is an important design choice as it allows for the larger system to survive, even when the underlying blockchain is compromised. Currently, Blockstack core developers decide which underlying blockchain(s) to support in which version of the software. Individual applications can decide to run the software version of their choice and keep their namespace on a particular blockchain, if they prefer not to migrate.

Ability to Construct State Machines: A key contribution of Blockstack is the introduction of a logically separate layer on top of a blockchain that can construct an arbitrary *state machine* after processing information from the underlying blockchain. This *virtualchain* treats transactions from the underlying blockchain as inputs to the state machine; valid inputs trigger state changes. At any given time, where time is defined logically by the block number, the state machine can be in exactly one global state. Time moves forward as new blocks are written in the underlying blockchain, and the global state is updated correspondingly.

A virtualchain enables the introduction of new types of state machines, without requiring any changes to the underlying blockchain. This approach is especially beneficial for a new and developing technology. Introducing new state machines directly in a blockchain would otherwise require peers to upgrade, and upgrades potentially break consensus and cause forks. In practice, they are difficult to orchestrate [4]. Currently, Blockstack provides a state machine that represents the global state of a naming system, including who owns a particular name and what data is associated with a name. Further, it's possible to use the *virtualchain* concept to define other types of state machines as well.

3.3 Blockstack Layers

Blockstack introduces new functionality on top of blockchains by defining a set of new operations that are otherwise not supported by the blockchain. Blockstack has four layers, with two layers (blockchain layer and *virtualchain* layer) in the control plane and two layers (routing layer and data storage layer) in the data plane.

3.3.1 Layer 1: Blockchain Layer

The blockchain occupies the lowest tier (Figure 2), and serves two purposes: it stores the sequence of Block-

```
$ORIGIN werner.id
$TTL 3600
_http._tcp URI 10 1 http://54.231.237.47/werner.id
```

Figure 3: Example zone file.

stack operations and it provides consensus on the order in which the operations were written. Blockstack operations are encoded in transactions on the underlying blockchain.

3.3.2 Layer 2: Virtualchain Layer

Above the blockchain is a *virtualchain*, which defines new operations without requiring changes to the underlying blockchain. Only Blockstack nodes are aware of this layer and underlying blockchain nodes are agnostic to it. Blockstack operations are defined in the *virtualchain* layer and are encoded in valid blockchain transactions as additional metadata. Blockchain nodes do see the raw transactions, but the logic to process Blockstack operations only exists at the *virtualchain* level.

The rules for accepting or rejecting Blockstack operations are also defined in the *virtualchain*. Accepted operations, for example a name registration operation on a name that has not been registered by anyone yet, are processed by the *virtualchain* to construct a database that stores information on the current global state of the system along with a history of previous states at earlier blockchain blocks.

3.3.3 Layer 3: Routing Layer

Blockstack separates the task of *routing* requests (i.e., how to discover data) from the actual storage of data. This avoids the need for the system to adopt any particular storage service from the onset, and instead allows multiple storage providers to coexist, including both commercial cloud storage and peer-to-peer systems.

Blockstack uses *zone files* for storing routing information, which are identical to DNS zone files in their format. Figure 3 shows an example zone file. The virtualchain binds *names* to their respective *hash(zone file)*. While these bindings are stored in the control plane, the zone files themselves are stored in the routing layer. **Users do not need to trust the routing layer** because, as long as the zone file data is available, the integrity of zone files can be verified by checking their hash from the control plane. Currently, all zone files are public and globally resolvable and we plan to support private data linked to public zone files in the future.

In Blockstack's current implementation, nodes form a DHT-based peer network [9] for storing zone files. The DHT only stores zone files if their hash was previously announced in the blockchain. This effectively whitelists the data that can be stored in the DHT. Due to space constraints, we omit most details of our DHT storage from this article; the key aspect relevant to the design of Blockstack is that routes (irrespective of where they are fetched from) can be verified and therefore cannot be tampered with. Further, most production servers maintain a full copy of all zone files since the size of zone files is relatively small (4KB per file). Keeping a full copy of routing data introduces only a marginal storage cost (24MB as of June 2016) on top of storing the blockchain data.

3.3.4 Layer 4: Storage Layer

The top-most layer is the storage layer, which hosts the actual data values of name-value pairs. All stored data values are signed by the key of the respective owner of a *name*. By storing data values outside of the blockchain, Blockstack allows values of arbitrary size and allows for a variety of storage backends. **Users do not need to trust the storage layer** because, as long as the zone file data is available, they can verify the integrity of the data values in the control plane.

There are two modes of using the storage layer and they differ in how the integrity of data values is verified; Blockstack supports both storage modes simultaneously.

(a) Mutable Storage is the default mode of operation for the storage layer. The user's zone file contains a URI record that points to the data, and the data is constructed to include a signature from the user's private key. Writing the data involves signing and replicating the data (but not the zone file), and reading the data involves fetching the zone file and data, verifying that $hash(zonefile)$ matches the hash in Blockstack, and verifying the data's signature with the user's public key. This allows for writes to be as fast as the signature algorithm and underlying storage system allows, since updating the data does not alter the zone file and thus does not require any blockchain transactions. However, readers and writers must employ a data versioning scheme to avoid consuming stale data.

(b) Immutable Storage is similar to mutable storage, but additionally puts a TXT record in the zone file that contains $hash(data)$. Readers verify data integrity by fetching the data and checking that $hash(data)$ is in the zone file, in addition to verifying the data's signature and the zone file's authenticity. This mode is suitable for data values that don't change often and where it's important to verify that readers see the latest version of the data value. For immutable storage, updates to data values require a new transaction on the underlying blockchain (since the zone file must be modified to include the new hash), making data updates much slower than with mutable storage.

3.4 Naming System

Blockstack uses its four tiers to implement a complete naming system. Names are owned by cryptographic addresses of the underlying blockchain and their associated private keys (e.g., ECDSA-based private keys used in Bitcoin [4]). As with Namecoin, a user *preorders* and then *registers* a name in two steps in order to claim a name without revealing it to the world first and allowing an attacker to race the user in claiming the name. The first user to successfully write both a preorder and a register transaction is granted ownership of the name. Further, any previous preorders become invalid when a name is registered. Once a name is registered, a user can *update* the name-value pair by sending an update transaction and uploading the new value to the storage layer, changing the name-value binding. Name *transfer* operations simply change the address that is allowed to sign subsequent transactions, while *revoke* operations disable any further operations for names.

The naming system is implemented by defining a state machine and rules for state transitions in the *virtualchain*. Names are organized into *namespaces*, which are the functional equivalent of top-level domains in DNS—they define the costs and renewal rates of names. Like names, namespaces must be preordered and then registered. Expired names can be re-registered and names can be *revoked* such that they cannot be re-registered for a certain period of time.

3.4.1 Pricing Functions for Namespaces

Anyone can create a namespace or register names in a namespace, as there is no central party to stop someone from doing so. *Pricing functions* define how expensive it is to create a namespace or to register names in a namespace. Defining intelligent pricing functions is a way to prevent "land grabs" and stop people from registering a lot of namespaces or names that they don't intend to actually use. Blockstack enables people to create namespaces with sophisticated pricing functions. For example, we use the *.id* namespace for our PKI system and created the *.id* namespace with a pricing function where (a) the price of a name drops with an increase in name length and (b) introducing non-alphabetic characters in names also drops the price. With this pricing function, the price of $john.id > johnadam.id > john0001.id$. The function is generally inspired by the observation that short names with alphabetics only are considered more desirable on namespaces like the one for Twitter usernames. It's possible to create namespaces where name registrations are free as well. Further, we expect that in the future there will be a reseller market for names, just as there is for DNS. A detailed discussion of pricing functions is out of the scope of this article, and the reader is encouraged to see [8] for more details on pricing functions.

Like names, namespaces also have a *pricing function* [3]. **To start the first namespace on Blockstack, the *.id* namespace, we paid $10,000 in bitcoins to the network. This shows that even the developers of this decentralized system have to follow Blockstack rules and pay appropriate fees.**

4 Conclusion

We have presented Blockstack, a blockchain-based naming and storage system that can be used as a general-purpose "trust layer" for building other service and applications without relying on any third parties. Blockstack introduces separate control and data planes, and by doing so, it enables the introduction of new functionality without modifying the underlying blockchain. This is counter to prior designs, which typically involved the introduction of a new blockchain and cryptocurrency in order to introduce new functionality.

The design of Blockstack was informed by a year of production experience from one of the largest blockchain-based production systems to date. We have introduced several innovations for blockchain services, including the ability to do cross-chain migrations, faster bootstrapping of new nodes, and keeping data updates off the slow blockchain network. These improvements all make it easier to build new decentralized services using existing, publicly-available blockchain infrastructure.

References

[1] A next-generation smart contract and decentralized application platform, 2016. `https://github.com/ethereum/wiki/wiki/White-Paper`.

[2] Adam Back. Hashcash - A Denial of Service Counter-Measure. Tech report, 2002. `http://www.hashcash.org/papers/hashcash.pdf`.

[3] M. Ali, J. Nelson, R. Shea, and M. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *Proceedings of the 2016 USENIX Annual Technical Conference (ATC'16)*, June 2016.

[4] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 104–121, 2015.

[5] Coindesk. State of blockchain Q1 2016, May 2016. `http://www.coindesk.com/state-of-blockchain-q1-2016/`.

[6] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. Draft, ipfs.io, 2015. `https://github.com/ipfs/papers`.

[7] Jude Nelson and Larry Peterson. Syndicate: Virtual cloud storage through provider composition. In *ACM BigSystem*, 2014.

[8] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan. An empirical study of Namecoin and lessons for decentralized namespace design. *WEIS '15: Proceedings of the 14th Workshop on the Economics of Information Security*, June 2015.

[9] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from IPTPS*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Tech report, 2009. `https://bitcoin.org/bitcoin.pdf`.