# An empirical study of Namecoin and lessons for decentralized namespace design

Harry Kalodner*, Miles Carlsten*, Paul Ellenbogen*, Joseph Bonneau†, Arvind Narayanan*
*{kalodner,carlsten,pe5,arvindn}@cs.princeton.edu †jbonneau@gmail.com
Princeton University

## Abstract

Secure decentralized namespaces have recently become possible due to cryptocurrency technology. They enable a censorship-resistant domain-name system outside the control of any single entity, among other applications. Namecoin, a fork of Bitcoin, is the most prominent example.

We initiate the study of decentralized namespaces and the market for names in such systems. Our extensive empirical analysis of Namecoin reveals a system in disrepair. Indeed, our methodology for detecting "squatted" and otherwise inactive domains reveals that among Namecoin's roughly 120,000 registered domain names, a mere 28 are not squatted and have nontrivial content. Further, we develop techniques for detecting transfers of domains in the Namecoin block chain and provide evidence that the market for domains is thin-to-nonexistent.

We argue that the state of the art in mechanism design for decentralized namespace markets is lacking. We propose a model of utility of different names to different participants, and articulate desiderata of a decentralized namespace in terms of this utility function. We use this model to explore the design space of mechanisms and analyze the trade-offs.

## 1 Introduction

**Decentralized namespaces.** We initiate the study of *decentralized namespaces* from an economic perspective. A namespace, as we define it, is an online system that maps names to values. The Domain Name System (DNS) is the most prominent example. A web service such as Twitter that allows users to claim usernames and create profiles can be thought of as implementing a namespace. To be memorable by humans, namespaces must support arbitrary user-chosen strings as names. To be secure, namespaces must map each name to the same value for all users; adversaries shouldn't be able to convince a user that any other value is correct.

The problem of *decentralized* namespaces has long been recognized as an important one. The DNS is a critical yet centralized component of the Internet and those who control it can alter the web for all users. The controversies around the shutdowns of `wikileaks.org` and the 2011 domain name seizures by the U.S. DoJ and DHS illustrate why many researchers and activists have sought decentralized alternatives [1].

The above three properties — security, user-chosen names, and decentralization — are known as *Zooko's triangle* [2]. Until 2011, designing a system that exhibited all three was conjectured to be *impossible* [3]. The rationale was that enforcing the uniqueness of name-value mappings and a consistent view of the directory for all participants would require a centralized server or a hierarchy.

**Cryptocurrencies and Namecoin.** Cryptocurrency technology enables building a namespace with all three properties. Put simply, the block chain is a global, distributed data structure that can be repurposed as a directory. *Miners* execute a consensus protocol to establish the state of the system and are incentivized to do so by mining rewards they receive in exchange for their participation. As long as a majority of miners — weighted by computing power — follow the protocol, all users will see a consistent view of the directory when they query it. This in turn gives the system, and hence

the underlying currency, economic value, making miners' actions profitable to them.

*Namecoin* is a cryptocurrency that realizes a decentralized namespace. It is the first altcoin from Bitcoin with its own block chain. It offers the same features as Bitcoin with the addition of a name/value store that can be used to hold arbitrary data (see Section 2). The name/value store supports various applications; primarily, Namecoin has been used for domain-name resolution for the '.bit' alternative TLD, and by the online identity service, OneName, which utilizes the Namecoin block chain to record data about its members.

Namecoin offers a novel solution to the technical challenges of decentralized namespaces. However, there are also economic challenges. These arise from the fact that although namespaces theoretically support infinitely many names, the supply of names that are memorable and meaningful to humans is scarce. Allocating these names to users is therefore a mechanism-design challenge. *The central thesis of this work is that this mechanism design challenge is far harder than realized.* A system that gets it wrong may "work" in a narrow technical sense, but may not be useful to real users.

Specifically, there are several crucial questions to consider: how do we model the economic behavior of the users of a namespace and what are the goals of mechanism design for namespaces? How well does Namecoin succeed at attaining these goals and what are its limitations? If Namecoin is not the ideal design, can we analyze the design space as a guide to the creators of future decentralized namespaces?

**Our contributions.** Along the above lines, we make the following contributions. We begin by proposing a model of utility of different names to different participants and articulating desiderata of a decentralized namespace in terms of this utility function (Section 3). We highlight the difficulty of mechanism design even in a toy model and explain the importance of making the model more realistic by incorporating extensions such as time-varying preferences.

The central contribution of this paper is a thorough empirical analysis of the Namecoin ecosystem. We develop a series of criteria based on the block chain, network behavior, as well as content to distinguish active websites from parked or squatted names (Section 4). This allows us to iteratively filter our dataset of around 120,000 registered names in Namecoin, leaving a mere 28 that are not squatted, and have nontrivial content.

We then delve deeper into the economics of names. Namecoin has a built-in ability to transfer names, which is a secure way to trade them using the block chain. However, it is not obvious which transactions correspond to such sales, as opposed to regular name updates. We develop a novel analytic technique to distinguish the two types of transactions and find evidence for about 250 transactions in the entire history of Namecoin that may represent sales of names (Section 5). Of course, it is possible that more sales have occurred off the block chain, but there doesn't appear to be a widely known marketplace for Namecoin names.

Based on all the empirical evidence we present, we are left to conclude that the Namecoin ecosystem is dysfunctional. The vast majority of registered names represent squatting and there is little evidence of a secondary market for names. While there could be many factors that explain the lack of adoption, there appears to be clear room for improvements in the design to minimize squatting and other problems. To this end, in Section 6, we explore the design space of decentralized namespaces and make recommendations.

**Why study namespaces?** Although we find that the Namecoin ecosystem is in disrepair, studying namespaces is important. While there currently doesn't seem to be widespread dissatisfaction with today's DNS causing users to seek censorship-resistant alternatives, the existence of such alternatives provides a valuable hedge against a potentially abusive central authority. Besides, domain names are just one application of namespaces. Centralized directories for user public keys have fared much less well than DNS, and the service OneName, which we discuss in Section 8, is an interesting alternative. Yet other applications for namespaces such as control of digital assets have been proposed. And of course, the problems posed by namespaces are intellectually interesting to study. We think decentralized namespaces have many important applications, but their promise hasn't been realized so far. Our work helps understand why this might be and lays the groundwork for a more rigorous approach to building such systems.

## 2 Background: Namecoin

In this section we cover the background of Namecoin. We begin by looking at the history of the cryptocurrency and then explain why a block chain can be used for our definition of a namespace. We then proceed to discuss the technical design of Namecoin and the mechanisms used in the design. We conclude this section with a discussion about the applications of Namecoin.

**A note about the term 'namespace'.** In computer science, a namespace is simply a container for a set of names, so that names in a single namespace must be unique but the same name can exist in different namespaces. We have chosen to use the term in a related but different way: it's a *system* that includes client and server software, users, a mechanism, and so on. In fact, Namecoin contains namespaces in the computer science sense, which we term *subspaces* in this paper.

### 2.1 History

Namecoin is an alternative cryptocurrency, or altcoin, modeled after Bitcoin [4]. Furthermore, it is the first altcoin in the sense that it was the first to create its own block chain, separate from Bitcoin's. Namecoin shares many similarities with Bitcoin, including the same method for proof-of-work, the same coin cap, the same block creation time, and all of the same transaction operations (with a few additions). Namecoin was inspired after discussions about a BitDNS [5] protocol using a block chain to manage a domain name lookup service. The motivation was that a central authority managing domain names, such as ICANN, requires too much trust in a single entity and represents a single point of failure. The first Namecoin block was mined in April 2011, and as of this writing, over 215,000 total blocks have been mined in the Namecoin system. Because of its similarities with Bitcoin, Namecoin was able to be merge-mined and has been merge-mined with Bitcoin since October 8, 2011 (see the next subsection).

### 2.2 Description of the block chain

Namecoin is minted and maintained by a decentralized peer-to-peer network.[1] Namecoin transactions require the digital signature of the account holder to prevent theft, and every transaction is published in an append-only hash chain, called the block chain. The block chain can be extended with new transactions by any participant, and such participants (known as miners) obtain newly minted Namecoin currency (NMC) and transaction fees from the transactions for performing this function. Extensions to the block chain require a proof-of-work that rate-limits the process (to approximately one extension every ten minutes) which enables a steady inflation rate, ample competition among participants to extend the block chain, and adequate time to obtain and verify the history of the block chain for new participants. Informally, the proof-of-work protocol in Namecoin is intended to maintain the following two essential properties about the block chain:

- Every party eventually agrees on the order and correctness of transactions in the block chain.
- Any party can publish a transaction (for a fee), which will then be verified and, if valid, included in the block chain within a small bounded delay.

One use case for a block chain is a tamper-evident log. That is, it can be used as a data structure that stores user-supplied data, and allows users to append data to the end of the log. Each new block has a hash of the previous block, so if data that is earlier in the log is altered, it will be detected. If an adversary wants to tamper with data anywhere in this entire chain, in order to keep the hash pointers consistent, he will have to tamper with the hash pointers all the way up to and including the current block. Thus it emerges that by just remembering the single hash pointer of the head of the chain, users essentially remember a tamper-evident hash of the entire list, all the way back to the genesis block. Namecoin, and all block chain based cryptocurrencies, use this tamper-evident log in order to record all the transactions between users.

**Block chain security.** The value and stability of a cryptocurrency are directly related to the amount of proof-of-work involved in the calculations of the blocks because this work is what keeps the data distributed and secure in the tamper-evident block chain. For both Bitcoin and Namecoin, the proof-of-work is shown by calculating the hash of a new block and a random nonce over and over until the

[1] This description is adapted from [6].

calculated hash has a certain number of leading zeros. The number of leading zeros required by the hash is referred to as the difficulty threshold. Namecoin enjoys a very high difficulty threshold for the proof-of-work because it is similar to Bitcoin and supports "merge mining" with Bitcoin. This means that miners who are mining Bitcoin can also choose to mine Namecoin at the same time with no extra work. Essentially, this is because the miner is using their computational power to solve a cryptographic puzzle that satisfies the proof-of-work for both block chains at the same time. This is advantageous for the miners because they are rewarded with coins from both systems, and helps Namecoin because it gives the Namecoin network a vastly increased amount of hash power over what it would have if it did not support merged mining. Including the merge miners, Namecoin has aproximately one third the hash rate of Bitcoin. This provides resilience to a 51% attack, although a sufficiently large Bitcoin mining pool could still execute this attack.

### 2.3 Technical details of names

In this subsection and the next, we present details of Namecoin, separating the technical solution from the mechanism design choices.

The feature that separates Namecoin from Bitcoin is that Namecoin is a namespace, and can be used to register name/value pairs that can be stored in the block chain and traded amongst individuals. This registration is done using the three script operations exclusive to Namecoin: `NAME_NEW`, `NAME_FIRSTUPDATE`, and `NAME_UPDATE`. In order to understand the registration process, we think it is helpful to walk through the registration process of `name`, roughly following Figure 1.

**NAME_NEW.** To start, the user will need to select a coin to be crafted into a token (or special coin) that represents a name and whose value can be changed by whoever possess the token. The next step to register a name is to make a transaction that uses the `NAME_NEW` script operation in a transaction sending the token from one of their addresses to another. Using `NAME_NEW`, a user can indicate an interest in `name` for a name/value pair by posting a hash commitment of the desired `name` in the scriptPubKey of the transaction. The reason the user posts the name in hashed form first, rather than in plaintext, is to prevent front-running, which we will explain in Section 6. The `NAME_NEW` operation acts as a signal in the block chain for name parsers to indicate that the next part of the scriptPubKey will be the hash commitment to a name.

**NAME_FIRSTUPDATE.** After doing this, and waiting for 12 or more blocks on top of the one containing the `NAME_NEW` transaction (to ensure that the block chain reaches consensus on the `NAME_NEW` transaction), the same user can use the output of the `NAME_NEW` transaction as the input for the `NAME_FIRSTUPDATE` transaction. Once completed, this will associate the chosen `name` with `value` selected by the user. Similar to `NAME_NEW`, `NAME_FIRSTUPDATE` allows data to be posted in the block chain as part of the scriptPubKey of a special transaction. To create a `NAME_NEW` transaction, a user will select, as input, the output of the `NAME_NEW` transaction. They will then use another address they control as the output for the transaction. The scriptPubKey of this transaction will contain a `NAME_FIRSTUPDATE`, the `name` desired, the random nonce used in the `NAME_NEW` hash commitment, and the first `value` for the name to take.[2] In order for this transaction to be valid, a miner will verify that the `name` and the provided nonce do, in fact, hash to the commitment in the appropriate `NAME_NEW` transaction. The output of this transaction now contains the token representing the name/value pair for `name` and `value`, and whoever can unlock and spend the output can utilize the final new operation, `NAME_UPDATE`.

**NAME_UPDATE.** The third and final new operation in Namecoin is the `NAME_UPDATE` operation. Again, this operation's arguments (the `name` and `newValue`) are stored in the scriptPubKey of a special transaction. This transaction must have as input a `NAME_FIRSTUPDATE` or `NAME_UPDATE` output with the same `name`. This operation has three uses: updating, renewing and trading a name. If the user wants to change the `value` associated with a given `name`, they will update `name` with this operation, providing a `newValue`. If names can expire, as they do in Namecoin, then this operation can also be used to renew a name by providing a `newValue` that is the same as the old `value`.

[2]There are also other operations that exist for reasons of compatibility with Bitcoin; this seems to exist to minimize changes to Bitcoin's script handling code.

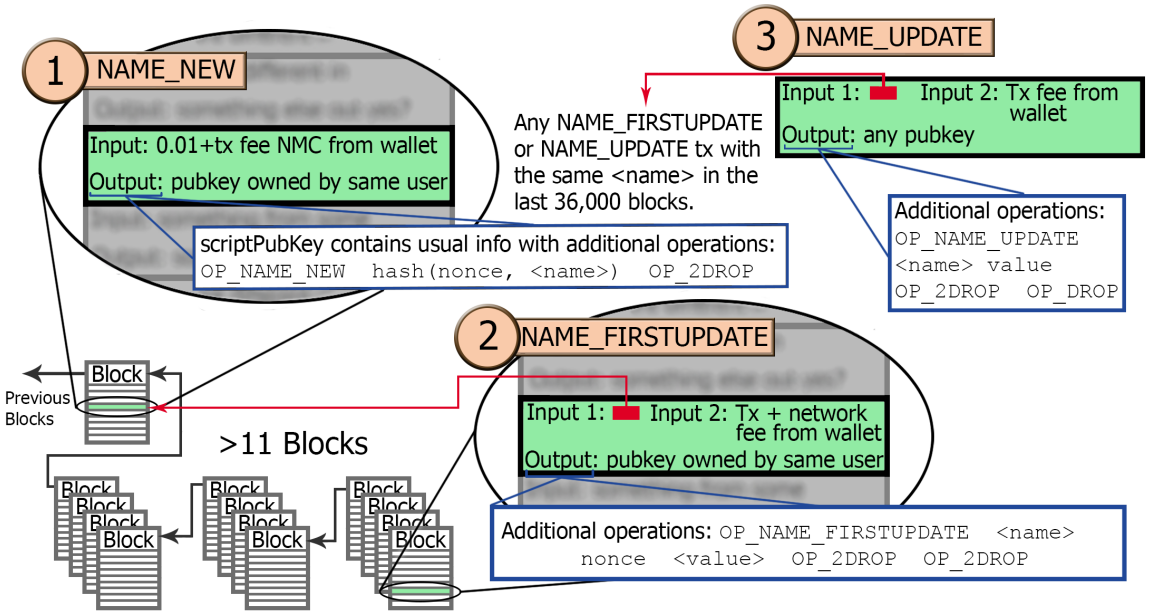


Figure 1: Namecoin name registration protocol described in Section 2.3

In either of these cases, the user will use an address they control as an output of the transaction. The final reason to make a NAME_UPDATE transaction is to trade the special coin to another user. In this case, the user will put, as an output, one of the other user's addresses instead of their own. Once the transaction resolves, the other user will have control over the special coin and can change the value to whatever they deem fit. Because the ownership of name is associated with the ownership of the special coin, if the buyer is paying for the name with Namecoins, the exchange between the payment and the name can be atomic (meaning they happen in the same transaction and either are only valid if the other is as well).

## 2.4 Mechanism design

**Fees.** Namecoin has been implemented with various fees and protocols to incentivize the behaviors of the users. The special token used in the NAME_NEW transaction has a value of 0.01 NMC. This coin will be not be spendable like other Namecoins while it has a name attached to it. For all of the transactions, NAME_NEW, NAME_FIRSTUPDATE, and NAME_UPDATE, the default behaviour is to have the user pay a transaction fee to the miner. The current expected transaction fee, which is programmed into the Namecoin client, is 0.005 NMC on each transaction. Historically, Namecoin also had a network fee attached to the NAME_FIRSTUPDATE transaction. The network fee is different from the transaction fee; the transaction fee is paid to the miners, whereas the network fee was destroyed (with an OP_RETURN) when a NAME_FIRSTUPDATE transaction was confirmed. The network fee varied over time — it started at 50 NMC at the genesis block, but decreased by a factor of 2 every 8192 blocks (which is approximately 2 months). The purpose of the network fee was to have a large initial cost to claiming names to deter users from quickly claiming all the desirable names, but then decay off so that eventually the cost of registering a name becomes negligible. As of block 85585, the network fee became small enough that it rounds to 0 and is no longer added onto the transaction. The current implementation of Namecoin has no fees other than the transaction fees and the investment of the token coin.

**Expiration.** Namecoin has an expiration time for names. Originally, the time period for a name to expire was 12,000 blocks, but by March 2012, the expiration period was increased to 36,000 blocks (which comes out to about 250 days). If a particular name hasn't been mentioned in a NAME_FIRSTUPDATE or NAME_UPDATE in 36,000 blocks, the name

becomes available again for any user to claim with NAME_NEW and NAME_FIRSTUPDATE. Similarly, a NAME_UPDATE must cite a NAME_FIRSTUPDATE or NAME_UPDATE that is less than 36,000 blocks old as input.

**Changing the protocol.** Changing the protocol in a cryptocurrency requires a "hard fork" or a "soft fork" depending on the extent of the change. In a mature system like Bitcoin, this is very difficult. In a fledgling system like Namecoin, however, it is much easier has happened multiple times. Both the addition of merge mining with Bitcoin, and the increase in expiration length were not initially in the design of Namecoin and required hard-forking changes. In order to make these changes, the Namecoin community decided on arbitrary blocks at which point the new protocol would be enforced. Explicitly, up to block 19,199 blocks that were merge mined with Bitcoin were not allowed into the Namceoin block chain, but starting on block 19,200 they were.

### 2.5 Applications

There are many different subspaces in Namecoin, and the different subspaces correspond to applications. When claiming a name, a user prepends the name with a subspace ID and a slash. Namecoin was created to be very general so that it would be useful for any application that would benefit from an online name/value store. While d/ has the most registered names, there are many used subspaces in Namecoin. The separation of namespaces is not enforced by the protocol in any way, but merely agreed upon by consensus of all users, analogous to open web standards.

**.bit Domains.** The vision for Namecoin was to use one of these subspaces for DNS lookup in the .bit TLD. Explicitly, Namecoin names associated with .bit domains are prepended with the subspace ID d/. If a user wanted the domain example.bit, they would claim the name d/example. The owner of example.bit would then set the value to their server address in a way that would be understood by .bit compliant DNS servers as described in the .bit specification [7].

Most major web servers, such as Apache, ngnix, and lighthttpd will accept connections through .bit domains with minor modifications to their per-site configuration files.

There are a number of ways to resolve .bit domains to IP addresses with varying levels of security. The most secure option is running local DNS resolution software. The three major projects created for this purpose are NMControl [8], a local DNS server, FreeSpeechMe [9], a Firefox addon, and DNSChain [10], another browser extension. NMControl and FreeSpeechMe both hold a local copy of the block chain and query it for values associated with .bit domains. They parse the value of the name using the specification and seamlessly direct the user to the resolved IP address.

There are a couple less secure options to access .bit domains. Users can connect to specific DNS servers such as OpenNic [11] which support resolving .bit domains. However this requires trusting that the DNS server is working properly and accurately reporting results. A final option is using a proxy server hosted in a standard TLD. This requires no local configuration, but users must go to example.bit by visiting example.bitproxy.com.

**OneName.** The name/value data store in Namecoin has applications beyond DNS. OneName is an online identity service that runs on top of Namecoin, making use of the data store. OneName is a centralized service, but one can use other clients to interact with the block chain in a way that's interoperable with OneName. The idea behind OneName is that a user can have a name/value pair in the block chain that associates said name with different online identities such as an email, GitHub, Twitter, and Bitcoin address. A OneName user can then confirm ownership of accounts on any of these services by referencing their OneName name through some messaging channel in each respective system. For example, if Alice wants to tie her twitter username to her OneName username, she must tweet the message "Verifying that +Alice is my openname (my Bitcoin username). https://onename.com/Alice". This is similar to the verification scheme Keybase [12] uses. A OneName account also has an associated Bitcoin address so users can easily find an address to use to send Bitcoins to a particular individual, if they want.

In order to make a OneName identity, a user can visit OneName's website to create an account. A user makes an account by selecting a username and then optionally entering in their actual name,

their account names, a short biography and picture, and a Bitcoin address. OneName takes the information given to it and automatically puts it into a name/value pair, with the name equal to the selected username, and the value containing all the other data. If the username is not already claimed by some user in the Namecoin subspace `u/`, then OneName posts this pair into the block chain. If it is already taken, the website will ask the new user to select a different username. If a user so desires, they can alternatively manually create a name transaction using the Namecoin client to create a name value pair with a name in the `u/` subspace and a value formatted to match the OneName specifications. To incentivize using their website interface, OneName covers the Namecoin fees for their users.

# 3 Modeling namespaces

**Names are scarce.** Many types of systems map keys or names to values. In his essay introducing his triangle, Zooko considers naming systems that include a system mapping PGP key fingerprints to keys. Fingerprints are hashes of keys. This would not qualify as a namespace by our definition because users cannot pick arbitrary names — that would require finding the hash pre-image of an arbitrary name (fingerprint), a hard problem by the definition of a hash function. Zooko (and similar prior work) considers such systems to lack human memorability of names. We replace this criterion with that of user choice of names, which is roughly equivalent but easy to define rigorously.

In a system that maps key fingerprints to keys, names are fungible and essentially infinite, and thus not scarce and have no market value. On the other hand, in any system that we call a namespace, scarcity is an almost inevitable consequence of free user choice. Even when names are scarce, the system architecture can make names more or less valuable. For example, usernames on Twitter are more valuable than on Facebook — the username or handle is a relatively important way to find a user on Twitter, whereas on Facebook it is done more often by navigating the social graph.

The scarcity of names makes the design of namespaces challenging. But it also makes the problem particularly amenable to cryptocurrency-based solutions. Since names have market value, and participation in the primary market requires the associated cryptocurrency, the currency becomes valuable. This incentivizes miners, making the system secure.

**The (non) role of trademarks.** Scarcity is also an issue in centralized namespaces, but it manifests in different ways. One key goal of most centralized systems is trademark protection. The trademark system can be seen as a legal mechanism to address the problems of name scarcity and limits of human memorability in the real world, independent of any particular technological system. Most centralized systems support some form of arbitration or dispute resolution in which trademark plays a role.[3] In a decentralized system, on the other hand, there is no easy way to enforce any rules that cannot be encoded algorithmically. As of yet there is no way to cryptographically assert that one is the owner of a trademark, and this may be impossible given the complexity and nuance of modern trademark law.

**Primary and secondary markets, algorithmic agents.** We use the term primary market to mean the part of the market where new names are issued to users for the first time. By contrast, the secondary market deals with the sale of a name already in use.

Whom do names "belong to" before they are sold on the primary market? To answer this, we must understand decentralized agents. Any cryptocurrency can be thought of as an algorithmic agent executed as a global, distributed computation. The agent has no capability to store private information but it can hold funds and transact with users of the system. Bitcoin encodes a very simple agent whose only function is to release currency into circulation at a pre-specified rate.[4]

In a nutshell, the agent implemented by a namespace initially owns all names and sells them to users. As we'll see in Section 6, there are many variants including whether names are sold or leased, whether or not they can be bought back, how names are priced, what the agent does with the funds received as payment, and so on.

[3] For example see ICANN's Uniform Domain-Name Dispute-Resolution Policy [13].

[4] At the other extreme, altcoins such as Ethereum allow any user to create an agent by making the central agent "Turing complete," that is, flexible enough to execute arbitrary programs specified by users on their behalf.

The most straightforward implementation of a secondary market is to leave it entirely external to the system. This is the approach taken by Namecoin, but again, a variety of choices are possible. We elaborate in Section 6.

**Squatting.** Even though names are scarce, it's not obvious what "squatting" means. After all, material goods are scarce, but we don't usually characterize car owners as squatting on them. The difference is that names have (vastly) different utility to different users. A squatted name is one that is owned by a user whose utility for that name is close to zero (or very small compared to the user who has the most utility for that name), purchased in hopes of selling to another user whose utility is higher.

Is squatting a problem for the system? This is a tricky question. If squatting exists merely because one side of the primary market is an algorithm and doesn't charge market price, then squatters could be seen as analogous to brokers or ticket scalpers. Such agents are sometimes considered to perform a useful function as market makers [14], or at least tolerated, but not considered an existential threat to the functioning of the market.

On the other hand, squatters can be viewed as analogous to land speculators. This analogy is supported by the fact that names, like land but unlike tickets, are not fungible. The speculator may have zero utility for a name and makes no use of the name himself; he hopes that demand for a name will rise in the future, and may therefore not sell to a buyer who has a positive utility today. This could lead to a market failure in a few ways. The uncertainty around future demand means that some names may be squatted indefinitely, while legitimate users may end up with sub-optimal names. If most valuable names are locked up by squatters, it may prevent the growth of the system, in turn preventing the growth in market price for names that speculators are hoping for. Land speculation has also been criticized as contributing to a market failure [15].

In this view, squatting may exist and may be problematic even if the primary market is able to discover the market price. But if the primary market is algorithmic, it only exacerbates the squatting problem.

**Complexity of the utility function.** Our argument above is essentially that the utility of a given name to a given user may vary with time. This is particularly relevant to cryptocurrency-based namespaces — in addition to the usual network effects that make the adoption of some new products challenging, cryptocurrencies need to "bootstrap," which presents an additional difficulty in getting off the ground. A cryptocurrency may not be sufficiently secure if it has too few users. Thus, many users may have a zero or negligible utility for all names until they decide that the namespace is secure enough and has enough users to be worthy of taking seriously.

Another aspect of complexity of the utility function is diminishing marginal utility. A user named John Smith may want either the name JohnSmith or john-smith but not necessarily both. Time-variation and diminishing marginal utility make the mechanism design problem very tricky.

Suppose, to the contrary, that utility functions are time invariant, and utility functions of a user for different names are independent. Then we may state the goal of the system as ownership of each domain by the user with the maximum utility (or willingness to pay) for that domain, as long as that value is above some threshold. We can easily realize this by making the primary market a second-price auction with a reserve price. No secondary market would be necessary.

Of course, this is oversimplified. Let's add diminishing marginal utility to this model. Suppose there are 10 John Smiths in the world and 10 variations of the name JohnSmith. A 10x10 matrix describes the utility of each user for each name. Further, none of these users has any utility for more than one name. Now we can state a welfare-maximization goal of finding the assignment of names to users that maximizes the sum of each user's utility for his assigned name, which translates to a maximum weighted graph matching problem. Or we could be happy with a Pareto-efficient assignment; since this is one-sided market (names aren't agents), this is the house allocation problem [16]. This shows that even in a highly simplified setting with a finite number of names and the same number of users, the mechanism design question is very tricky.

When we add time-varying preferences to the model, it is unclear if we can even formally state a goal for the system. The harder we make it for

a user to hold on to a purchased name, the easier it becomes for an adversary to "seize" a name (see below).

At any rate, the fact that preferences are time varying appears to be part of the reason that names in Namecoin expire and must be renewed; in effect, they are leased rather than purchased outright.

**Seizures.** The Namecoin community considers it a major goal to prevent "seizures" of domain names by adversaries [17]. To prevent seizures, then, it must be easy to hold on to a name after buying it. This is in tension with preventing speculation, which works by buying a name and holding on to it despite other people wanting it.

The reason it is even technically possible to prevent such censorship in the face of a well-funded adversary is that the adversary can't possibly preemptively buy up all the names that the victim might use. For example, Wikileaks might find any name with the string "wikileaks" in it to be acceptable, even if not ideal, for their purposes. In other words the victim's utility function has a large support; the adversary's utility for a name is *contingent on* the victim owning that name.

# 4 Analysis of .bit domains

In this section we analyze the current state of .bit domains. We begin at the highest level, looking at the repetition of values in order to detect squatters. Next we look at how many .bit domains are set up with query-able values. Finally we actually visit these query-able names and analyze the content we find there.

## 4.1 Detecting squatters

We investigate the balance between squatters and other users in the .bit subspace. As we discussed in 3, squatting is a critical issue in any decentralized namespace. We look at Namecoin in order to get a realistic view on the proliferation of squatting.

Namecoin possesses the same pseudonymity properties as Bitcoin and thus it is difficult to group names by their owner. Names are owned by individual addresses rather than by identities and it is a common practice to keep each purchased name under a different address. Because of this, it is difficult to assess how many names any single entity owns.

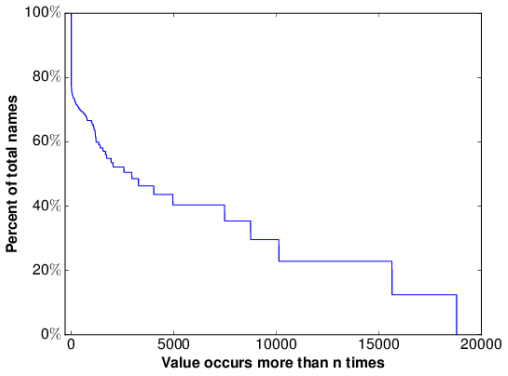


Figure 2: Analysis of squatting. Names whose values occur more than about 10 times can be safely considered to be squatted. In fact, the graph shows that the majority of names are held by prolific squatters who control thousands of names.

However, many squatters are easily identified through the values they set for the names they own. Since there is no built-in functionality for the listing and sale of names in Namecoin, squatters use the values of names they own to display contact information. This information generally comes in the form of either contact information stored directly in the value of a name, or contact information stored at the IP address to which the name resolves. We observed that a squatter's contact information is generally constant among all the names they own. Thus, by measuring the extent to which values are duplicated on the block chain, we can estimate the ratio of squatters to genuine users.

Regular .bit domain owners are unlikely to use values that are highly replicated. The value of a domain points to an individual server and most are unique. Repetition may occur in this circumstance is when multiple names resolve to the same point. This could occur when a single DNS server is resolving a large number of names to different websites. However when we performed the content analysis described in Section 4.3, we found that this does not happen currently.

There is no definitive threshold for deciding exactly how often a value has to occur in order to assume a squatter has been detected. However even the coarsest grained approach displays a massive amount of squatting on the block chain. Of

| Type of resolution | Count |
|---|---|
| Nameserver URL | 3200 |
| Nameserver IP | 148 |
| Single IP | 5848 |
| Multiple IP | 2 |
| Single IPv6 | 2 |
| Multiple IPv6 | 1 |
| Tor | 9 |
| Alias | 23 |
| Only subdomains | 2 |
| Total | 9354 |

Table 1: Comprehensive list of .bit domain resolution methods currently in use.

| Criteria applied | Count |
|---|---|
| Total Names | 196,023 |
| Valid DNS | 9354 |
| Curlable | 5374 |
| Not squatter | 745 |
| Without duplicates | 455 |
| Without errors | 278 |
| With content | 222 |
| Without ICANN hostname | 28 |

Table 2: Number of .bit domains which resolve to real content

the 196023 currently active names, there are only 34361 unique values.

In Figure 6 we examine the fraction of current .bit names that are squatted. We sorted all of the values that occur by the number of times they occur. The graph has a very sharp initial drop as we remove all names with values that occur only a few times. This leaves the majority of domains to drop off very slowly as we remove bigger and bigger squatters. Using a cut-off of $n = 10$, it appears safe to say that at least 76% of .bit domains are held by squatters.

### 4.2 DNS records

We now examine how .bit domains are used by genuine users. Only 9354 out of the 119624 .bit domains make any attempt to resolve to an IP address. Out of these a variety of types of IP resolution are employed.

In order to support DNS lookups, Namecoin provides a specification that allows for the support of most DNS record types. A name's configuration is stored in a JSON dictionary object which is placed in a name's value. Domain's can be configured in a number of different ways. The main methods are directly setting one or more IP (or IPv6) addresses or setting one or more secondary nameservers which hold information about a domain. Additionally records can link a name to a number of hidden-services URL schemes like .onion [18] and .i2p [19].

Namecoin supports a number of different name resolution methods. Different methods have different properties and thus it is interesting to inspect how people are setting up their domains. The vast majority of names point directly to an address. This includes people using IP, IPv6, and Tor. This is by far the most privacy preserving method since the IP address is drawn directly from the block chain. A client can simply look up a name in the block chain and immediately connect to the server. However, this privacy comes at the cost of flexibility since the server is directly encoded in the block chain and can not be changed without an update.

The much more flexible configuration, also commonly used, is name server delegation. Rather than directly listing an IP address in the block chain, one or more name servers are listed. This way a name owner can update their IP address without modifying the block chain. However this is an insecure delegation since Namecoin can not enforce any properties regarding the action of the external name server. The server will have full control over its interaction with users including the ability to track lookups or return results which aren't globally consistent.

### 4.3 Domain content analysis

After understanding how people connect their names with IP addresses, we explored what sort of content is reachable through .bit domain names. We attempted to download the front page of each of these 9354 domains over port 80 (HTTP). 3881 of these domains were unreachable or didn't serve web content, leaving us with only 5374 responsive domains.

Looking at the content of the servers' responses, we found that of the responsive 5374 domains, 4629 are owned by 3 different squatters. These do-

mains serve nothing of value and caused massive inflation in our reachable domain count.

Removing the squatters, we count 745 viable domains. However, many of these pages are mirrors or duplicates of each other. After removing 290 such duplicates, 455 domains remained.

A large number of pages were either error responses from the server or default pages from various web servers. Neither of these can be considered useful content. There were 177 such domains, and removing them left us with 278 domains.

Out of the remaining sites, many had very small amounts of content consisting of only a few words on a blank page like, "Welcome to mysite.bit." These pages, though valid uses of Namecoin, provide minimal utility to a visitor. Thus, we decided to look at only pages consisting of 15 or more words and images which brought us down to 222 domains.

These 222 domains make up the only websites reachable through .bit domain names which have any real content on them at the time of this analysis. We are further interested in the subset of these pages which don't come from a server that is accessible through a standard ICANN TLD as well. 83 of the pages directly redirect the user to a standard domain and 111 were manually identified as pointing to the same site as a standard domain. The domains which had ICANN hostname's along with their .bit hostname were manually identified and pinged to ensure matching IP addresses.

This left us with 28 pages serving non-trivial content that is uniquely available via .bit.

Our approach to analyzing content suffers from a few limitations. First of all we only queried the main domain over port 80, thus if any of the servers only respond to subdomains or only serve content over HTTPS, they are not included. Additionally when we detect how much content is on each page, we do not follow links and thus content could be hidden behind links. Despite these limitations we believe that we have missed very few legitimate domains.

### 4.4 Examining Squatter Name Choice

As we have seen, many individual squatters buy up very large numbers of domains. We now explore what sort of categories these names fall into and try to interpret the decisions these squatters have made. Here we explore two potential measures of name desirability, one using Alexa ranking, and one using the length of the name.

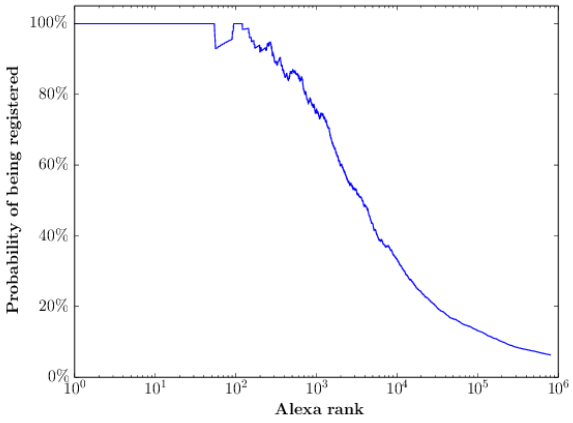


Figure 3: For each $1 \leq n \leq 1,000,000$, the probability that .bit versions of domains with Alexa rank approximately $n$ are registered.

We can see from Figure 3 and Figure 4 that there are patterns to the names that squatters have acquired. In Figure 3, we can see that squatters have claimed the majority of very high ranking Alexa sites.

In Figure 4 we see that there is a preference for shorter names. Table 3 shows that in fact all one- and two-character names are taken. It is not until we get to names with at least three characters that there are names available to register. Beyond three characters, the combinatorial nature of name strings makes it infeasible to register all names of that length. There are more names with a length of 4 available than there are total names registered currently.

Alexa ranking and name length are not an exhaustive description of squatter behavior. Future work would attempt to better model the valuation that squatters and other namespace participants put on names.

## 5 Secondary market analysis

Next, we seek to understand and quantify how names move between users. Our basic scenario is that Alice owns the name 'd/example' and Bob would like to purchase it from her. We explore various ways this sale can occur and how these sales can be detected.

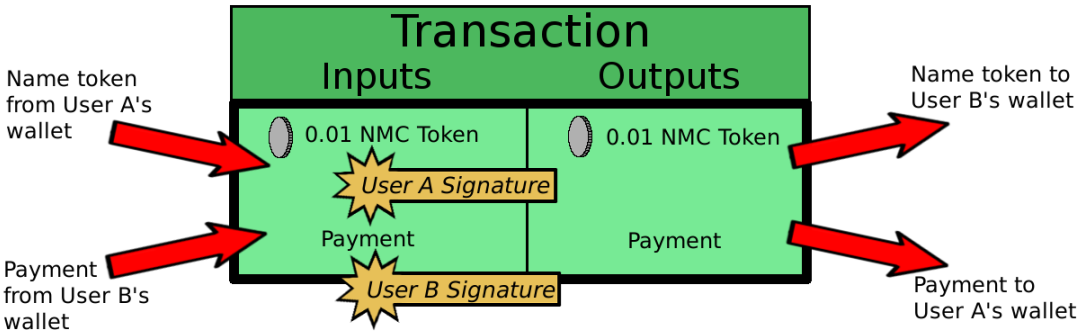


Figure 5: Anatomy of an atomic name transfer. Both the name and the payment are included in the same transaction, so one cannot fail to transfer to the other party without the other failing to transfer.

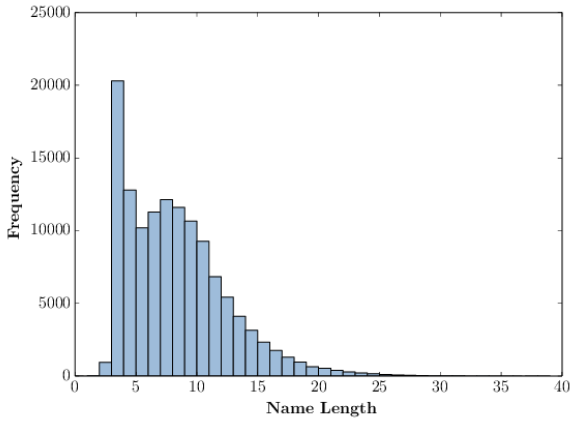


Figure 4: The number of Namecoin names registered as a function of name length.

| Name length | Percent registered |
|---|---|
| 1 | 100.00% |
| 2 | 100.00% |
| 3 | 58.61% |
| 4 | 1.00% |
| 5 | 0.02% |
| ⋮ | ⋮ |

Table 3: Percent of all domains of registered by length. All possible names are counted combinatorially using the rules given by [7].

### 5.1 Detecting atomic transfers

The safest way to buy and sell Namecoin names is through the use of atomic transactions. This is an important technique in cryptocurrencies whereby two parties can exchange digital assets (such as a name in exchange for currency) without a trusted intermediary without either worrying that the other will abscond after receiving her half of the bargain. We show how these transactions work in Figure 5. In an atomic transaction, Alice and Bob make their exchange in a single transaction. In the simplest form of atomic name transfer, Bob creates a transaction which transfers his payment to Alice and transfers 'd/example' to him. He then sends this transaction to Alice, the owner of the name, who verifies it, signs, and broadcasts the transaction to the block chain. Both Alice and Bob's signatures are locked to the inputs and outputs of the transaction so neither input can be spent individually without the full transaction. This transaction provides cryptographic security to both Alice and Bob since either both the name and coins will be exchanged or nothing will. Although there is nothing inherently different looking about this transaction on the block chain, there are a few possible techniques to detect them by implementation quirks.

The Namecoin client is a fairly underdeveloped piece of software and thus there is no built-in method of performing atomic transactions. In order to accomplish this task, the Namecoin RPC

client must be used from the command line. In order to simplify this task, a Namecoin developer created ANTPY [20], a piece of software to automate the creation of atomic transactions. This software has the quirk that the buyer's payment goes to the address that the seller held the name in. To find these transactions we queried the block chain for transactions with a `NAME_FIRSTUPDATE` or `NAME_UPDATE` input from the same address as a non name output. We then further reduced this set by eliminating transactions where the name stayed at the same address.

We searched throughout the history of the Namecoin block chain for transactions fitting this specification. Our query returned 13 transactions which we believe represent all transactions built by the ANTPY script. However this by no means represents all sales on the block chain. We next attempt to discover atomic transactions in a different way.

A implementation-agnostic method for detecting atomic name transfers is to find transactions that clearly use change addresses. This occurs when there are two non-name outputs in a transaction that has a name input. In this case the buyer did not want to pay all of his input to the seller and thus kept some for himself. This leaves the transaction with 3 outputs. Under normal circumstances, `NAME_UPDATE` transactions will only have two outputs, a name output and a change address. Thus every transaction with three outputs is very likely an atomic name transfer.

In the history of the Namecoin block chain, we found 6 transactions fitting this form. However 5 of the 6 were also detected by the previous (ANTPY) criterion.

The 14 atomic transactions which we detected are a lower bound for the number of name transfers. We are unable to query for all atomic transactions since if the buyer doesn't want any change from a purchase and the seller gives the buyer a new address to send payment to, the transaction is indistinguishable from a regular non-transferring name update.

## 5.2 Deriving an upper bound on number of sales

Following up on our lower bound from the previous section, we now derive an upper bound on the number of name sales using data from the block

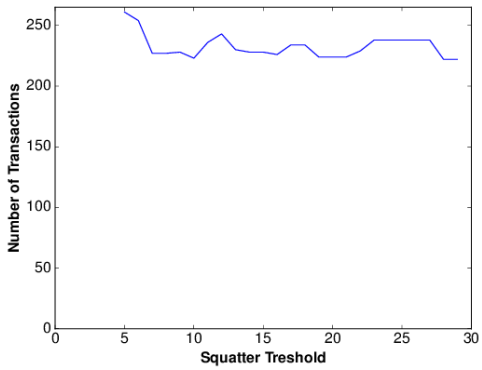


Figure 6: Number of transfers detected based on squatting threshold. For each $n$ on the x-axis ($5 \leq n \leq 25$), we plot the number of squatter $\rightarrow$ non-squatter transactions detected if we characterize names whose values occur $n$ or more times as squatted.

chain. Whereas atomic transactions, can (sometimes) be recognized simply from their contents, other name sale transactions are not recognizable. Here the payment could be a separate transaction or even made in a currency other than Namecoin. We would hope that we could detect changes in name ownership by looking at changes in which key owns a name. However, since the Namecoin client defaults to sending names to new addresses on update, there is no way to look at an update and tell whether or not a name is being transferred between owners.

In order to detect non-atomic transactions we must expand our view to the prior value of a name being updated. Certainly if the value does not change then the transaction is simply renewing the name, not transferring it. However considering all other transactions to be name transfers is far too conservative of a criterion. Users freely update the values of names whenever information in them becomes outdated.

Although there doesn't seem to be a way to detect non-atomic name transfers generally, there is an important subclass of these transactions which can still be detected — transfers from squatters to regular users. In the previous section we discussed our detection of squatters in the block chain which produces a list of values which with a high probability belong to squatters. Detecting transfers from squatters by finding names that change from one of

these values to a value outside this set gives us a strategy to detect transfers.

We employ an additional criterion to eliminate false positives to tighten our upper bound. If a name's value includes an info or email field and that stays the same in the updated value we can assume this is simply an update by the squatter.

Applying this analysis at various squatter threshold values, we see that the total number of squatter $\rightarrow$ non-squatter transactions detected holds at approximately 250 transactions. We emphasize that this value is likely an upper-bound since our criteria for reducing the number of transactions were quite conservative.

To summarize, we would expect that given the high percentage of squatted .bit names, if there is a flourishing secondary market it would be dominated by sales from squatters to regular users. Yet we are able to upper bound the number of such transfers to about 250, a tiny fraction of the number of squatted names. Further, even though Namecoin supports a secure way to transfer names, we find strong evidence based on known tools supporting this functionality that its usage is very low.

# 6 Exploring namespace design choices

Throughout this section we will refer to the concept of the decentralized algorithmic agent that we introduced in Section 3. Several of the options we explore will require this agent to implement rather complex algorithms. We will not directly address the question of practical feasibility of complex decentralized agents. We note, however, that based on proposed designs, these agents can be surprisingly powerful; we refer the reader to [6, 21, 22].

## 6.1 Control of names

At any point in time, each name is either controlled by some user or is unclaimed. In the latter case, it can be considered to be owned by the decentralized agent encoded into the cryptocurrency, and is on the primary market. Any user can attempt to purchase any name on the primary market. We will return later to the question of how the name is priced.

When a name is controlled by a user, how strong is that control? We can consider a hierarchy of increasingly weaker forms of control.

*1. Control lasts forever and names cannot be transferred.* It is straightforward to design a namespace in such a way that names are non-transferable (in Namecoin, it would be a matter of requiring the input and output of a `NAME_UPDATE` transaction to have the same address). Making names non-transferable would lock authority to update to the value restricted to the key used to first register the name. Such a system could deter speculative squatters because a speculator would not be able to sell a name to another individual. While it would be possible to transfer the private key that controls a name, the user receiving it would have no cryptographic assurance that the original owner does still posses a copy of the key. On the other hand squatters who wish to censor names or simply damage the system would be undeterred by the inability to transfer names.

As we noted in Section 3, utility functions are time-varying, so a technical restriction against name transfer goes against the logic of the market. It is possible that most names would be controlled by squatters who act as dealers and lease names to users. Essentially this would make the system a hybrid between a centralized and a decentralized namespace.

*2. Control lasts forever except if the user chooses to transfer or sell the name.* This approach is easy to understand as it is the most similar to physical property. A practical problem with this approach is that if a user loses her key, any names controlled by that user or key are lost forever. This may also happen if a user leaves the system, but this problem can be alleviated by having the primary market agent buy back names from users. This incentivizes the user to give up the name before leaving the system even if she can't find a seller for it. The agent essentially acts as an automated market maker.

*3. Names expire after a fixed period, except if the user renews it.* This is a practical choice that avoids the problems pointed out above, especially if the renewal fee is very small compared to the price of the name on the primary market. We expect that a rational user will keep renewing a name she controls unless her utility drops to near-zero, or she loses the associated private key and therefore

can't update the name. If the renewal fee is substantial, this becomes similar to a lease instead of ownership. The renewal fee is paid to the agent.

Finally, the agent may pay the user if she lets her name expires, returning her original fee to her. This is analogous to a deposit.

*4. Names always expire after a fixed period and return to the primary market.* Seizure or censorship becomes easy. Additionally, if we imagine that names appreciate in price by being used, just as land that is developed increases in value, this disincentivizes users from putting names to use since they have no expectation of being able to hold on to them. So this model is clearly inappropriate for applications like domain names, but perhaps useful in other contexts.

*5. Control over names may be preempted by other users.* This is an idea that has been proposed several times in the context of Namecoin, allowing users to either force a name to return to the primary market or even directly acquire it by paying the a fee to the agent and/or the current owner. This is intended to alleviate squatting, but it is hard to imagine this approach being superior to a functioning secondary market. On the other hand, the downsides are clear: seizures are now possible, and as before, uncertainty over future ownership may discourage legitimate use.

## 6.2 Markets and fees

**Primary market: auctions vs. algorithmic pricing.** There are essentially two choices for the primary market: auction and algorithmic pricing. Auctions are appealing, but there are practical problems due to the infinite number of names. For example, let's say an auction for a name is triggered whenever any user expresses an interest in buying that name. Since there are a potentially infinite number of names that a user is interested in, she would have to be able to participate in an auction at any time, which is a problematic assumption. Alternatively, names could go up for auction at fixed times, but this also has practical downsides.

On the other hand, names can be priced algorithmically. This is more applicable for applications like domain names than for personal namespaces. In the former case the utility of a name is less dependent on the user and therefore easier to price based on publicly available data. Several factors based on publicly available data can be incorporated into the pricing model: name length, frequency of the name (treated as a word) in text found on the web, traffic rank of the corresponding .com domain, and so on. Pricing can also vary with time; a blended model between auctions and algorithmic pricing would see initial prices decided algorithmically which decline steadily over time in a form of Dutch auction. Again the challenges in implementing this are practical: incorporating data external to the block chain into the computation of distributed agents is possible, but very complex and has not been successfully demonstrated in practice. If the secondary market is implemented within the system (see below), then the agent has access to pricing data for other similar names as well as for the same name if it was previously sold on the secondary market.

There is an interesting technical limitation of agents which affects algorithmic pricing. Since the agent doesn't have private storage, an adversarial node in the peer-to-peer network can intercept a user's request to the agent to purchase a name, and try to purchase that name ahead of the user, perhaps in the hope of turning around and selling it to the user. This is analogous to front running. In fact, front running is a problem in the current DNS as well, carried out by intermediaries that provide domain registry search services [23]. Fortunately, in the cryptocurrency context there's a simple cryptographic solution: the user first presents a sealed bid to the agent (here it is the name that needs to be sealed, not the price, which is public); after this message is confirmed in the block chain, the user (from the same address) reveals the name. Now the adversary cannot front run the user because he cannot spoof messages (transactions) from the user's address.[5]

Namecoin uses a simple model of a flat fee for all domains that varies with time (which we can think of as a trivial case of algorithmic pricing).

**Secondary market.** As long as the technical capability to transfer names exist, we would expect a secondary market to develop without any further design support. In addition, the ability to transfer

[5]This is the approach used in Namecoin, and the reason why `NAME_NEW` and `NAME_FIRSTUPDATE` need to be distinct operations separated by a 12-block waiting period, as described in Section 2.3.

names and payment in a single atomic transaction allows parties to transact without a trusted intermediary. This is straightforward to implement technically as we discussed in Section 5; Bitcoin and most altcoins including Namecoin support it.

However, there are benefits to the capability to post bids and offers on the block chain as well as execute trades. First, as a practical matter, when cryptocurrencies are in the bootstrapping stage, exchanges may not develop or may be fragmented. We see this problem with Namecoin. Secondary markets for DNS names are also fragmented. A fragmented or non-existent secondary market may in turn inhibit adoption of the namespace. Integrating an exchange (implemented, of course, in a decentralized form) may prevent this problem. As an interesting historical aside, early versions of Satoshi Nakamoto's Bitcoin code contained exchange functionality in prototype stage, but this was never rolled out [24]. Second, as mentioned above, executing trades via the block chain allows the agent direct access to price data, and this may help with algorithmic pricing. Finally, integrating an exchange allows shaping the secondary market, for example, by charging a fee for trades. As mentioned earlier, it is technically feasible to deter private trades; this would force or incentivize trades to happen through the integrated exchange.

### 6.3 The agent's finances

In a centralized namespace, the owner of the namespace has a clear goal of maximizing revenue from name sales. In a decentralized system, this is typically not a meaningful goal. After all, the agent is not a real-world entity. How, then, should the agent handle its finances?

Our first simplifying observation is that there is no point in the agent holding on to any funds — even if it needs to buy names from users, it can simply print money (introduce new coins into the system) as necessary. Of course, from a practical perspective it may be easier to have the agent store funds; this corresponds to the proposal in Namecoin to treat registration fees as deposits and to hold them in "escrow" [25]. Conceptually, however, we can get rid of this option.

That leaves two options. The first is to distribute its revenue to miners, in proportion to their hash power at the time of the transaction. This is technically straightforward to do by structuring payments as transaction fees to be collected by the miner who first mines a block containing that transaction. The other option is to "burn" the coins representing the payment, i.e., mark them as permanently unspendable. This is also technically trivial. Assuming that the market capitalization of the currency is unaffected by burning some of it, it has the effect of distributing the agent's revenue to all holders of the currency in proportion to their holding. Both of these appear to be reasonable options, and Namecoin has used both; it is not clear which is better in terms of incentivizing the long-term growth and security of the system.

## 7 Related Work

Here we discuss attempts besides Namecoin at decentralized cryptocurrency-based namespaces or domain-name systems and more broadly technologies that utilize the block chain.

Bitshares is another altcoin that includes proposals for a namespace as one of its goals [26]. Emercoin recently emerged as a Namecoin competitor [27].

More broadly, the ability to publish messages to the block chain immediately allows a variety of applications. Physical property, shares of stocks, bonds, or any other real asset can be digitally represented on the block chain. Overlay protocols such as Mastercoin and Colored Coins specify a syntax and semantics for such digital representations [28, 29]. CommitCoin allows for putting hash commitments on the Bitcoin block chain in order to timestamp data in a trustless manner [30].

There are a number of more complex applications that require additional primitives. Financial derivatives are contracts whose value depends, in some mutually agreed-upon way, on the price (or movements in price) of an underlying asset. Implementing derivatives of digital assets, then, requires user-defined logic (or scripts) for transaction validation. This can be accomplished via an altcoin with a flexible scripting language such as Ethereum [21]. Furthermore, since derivatives depend on prices, scripts that implement them require access to price feeds as input. Otherwise, it requires a trusted third party (known by a Bitcoin address) to regularly publish suitably encoded, signed price

statements reflecting external reality. These can be published directly to the block chain or distributed off-chain and only added to the block chain when needed to redeem an asset. More generally, such entities could publish any data feed representing news or other events.

Bitcoin can act as a platform for fair secure multiparty computation [31, 32, 33]. These are SMC protocols augmented with Bitcoin operations, e.g., a payment from one participant to another, or a deposit.

Finally, a set of related ideas known as decentralized autonomous organizations (among several other names) has stirred considerable excitement in the community recently. These combine several primitives discussed above — digital assets, long-lived scripts implementing arbitrary logic governing those assets, data feeds, and out-of-band communication. Some proposals for DAOs incorporate human input in various forms: one is a decentralized agent farming out computationally intractable tasks to humans [34]. Another is voting by shareholders of decentralized agents to enable modifications to the logic (i.e., script).

# 8 Discussion and Conclusion

Our empirical analysis of Namecoin and exploration of the design space reveal several potential mechanism design weaknesses. First, Namecoin's fees are far too low to deter squatters. The system utilizes neither an auction nor algorithmic pricing that might help price names near their market value. Worse, users who paid a high network fee in the early period and who wish to leave the system have no way to recoup any of their investment by returning the name to the primary market.

These problems are exacerbated by the lack of a functioning secondary market. Buying a squatted name, even if contact information is available on the block chain, is a cumbersome manual process. There is no way to, say, search for all names available for sale matching a given string and below a specified price limit. Namecoin does not integrate an exchange into its core functionality, nor have Namecoin developers chosen to build and promote one outside the system.

**Is a decentralized namespace for domain names viable?** Even with a hypothetical system where the above weaknesses are fixed, it's not clear that it would be a viable platform for a decentralized DNS. Namespaces are a solution to the problem of mapping names to values. However, in a domain name system, the problem that needs to be solved is to map *entities* — legal entities, brands, or identities — to addresses/values.

Naturally, mapping entities to names (or directly to values) is not a purely technical problem, but requires human judgment. In the traditional DNS, arbitration based on trademark law is a mechanism to prevent names that are confusingly similar from being controlled by different entities. This is augmented with Extended Validation certificates, which, when used, establish the legal identity of the name owner. With these mechanisms in place, the user must still provide a name to the system, but at least she can then verify that it corresponds to the correct entity she had in mind. This type of verification is much harder to incorporate in a decentralized system. We are not aware of any designs for decentralized systems that attempt to do so.

If such mechanisms are not developed, can decentralized namespaces for domain names be viable at all? Or will they depart so far from user expectations as to be unusable?

Other solutions for usability might be possible. Web search engines are used in practice as a way of mapping entities to names. Indeed, such "navigational queries" are one of the primary ways in which people use search engines.[6] Navigational queries are about equally viable in centralized and decentralized systems. Presumably, using search engines and following links from other sources (including bookmarks) will allow a user to gradually memorize an entity $\rightarrow$ name mapping.

Of course, if users *always* used these nagivational aids and relied on them entirely to find the correct name, then those names don't have to be human-memorable or user-chosen, so one doesn't need a namespace at all. Indeed, .onion domains force users to rely on navigational aids since names are public keys and can't be freely chosen by users. Anecdotally, however, it is clear that this results in poor usability and memorable names would be an improvement.

[6] Bing reports that about 30% of queries are navigational [35].

We remain neutral on the question of whether a decentralized namespace for domain names can be viable. But the human factors we discussed do complicate the mechanism design question. In Section 3 we proposed utility maximization as a (strawman) goal of the mechanism, and suggested that an auction would satisfy this goal. However, in an auction-based system, it may turn out that an adversary interested in phishing has a higher utility for (say) `bankofamerica.com` than Bank of America does. It is not clear how to design a mechanism that will discourage such outcomes.

**Comparison between .bit and OneName.** In contrast to the .bit subspace, the online identity service, OneName, has a very large user base. We found that after limiting the number of name/value pairs to only those with unique, well formed entries (using the strategies listed in section 5) OneName has roughly 20,000 pairs, which dwarfs the 1000 or so .bit domains. It is too early to tell whether or not these users are putting their OneName identities to any real use, but at least in terms of registration numbers, interest appears to be high.

Perhaps the biggest reason for this is that while scarce, OneName names are not nearly as valuable as domain names, and so it is far less important to get the mechanism design right, or have any mechanism at all. Unlike domain names that must be remembered by humans, users typically find OneName identities through links from other social media services or by searching for the individual's real name.

A second reason is that OneName benefits from being a centralized service on top of a decentralized protocol, OpenName (which, as of this writing, is implemented on top of Namecoin). If a name is squatted, a typical user who wants that name might not even know how to contact the squatter because they are trying to register a name through the OneName website which doesn't facilitate this process. While a user could technically interact with the 'u/' subspace of the block chain directly, most OneName users don't.

The .bit subspace, on the other hand, suffers from the lack of any centralized service providers coupled with the poor usability of the client software. A new user faces many significant hurdles in registering a name. The first step is acquiring a Namecoin client which requires downloading, verifying, and storing the entire block chain, which is currently about 1.6 GB. Then she must acquire NMC, which requires connecting the user's wallet on a cryptocurrency exchange (which supports Namecoin) to her bank account. In our experience, there is a 3-day wait after registering with an exchange and being allowed to transfer funds. Now the user is finally ready to register names, but she must also remember to update names every 250 days or run software in the background that will automatically handle this.

In conclusion, designers of decentralized namespaces must pay careful attention to mechanism design to deter squatters and facilitate a healthy secondary market. Further, the usability of the overall ecosystem, which includes not just users who own names but the applications that utilize the namespace for web browsing or other tasks, is paramount. Finally, a hybrid model of centralized services that utilize an underlying decentralized platform is worth considering.

**Acknowledgement.** We would like to thank Ed Felten and Muneeb Ali for helpful discussions and feedback. This work was supported in part by NSF Grant CNS-1421689.

# References

[1] Paul Mutton. WikiLeaks.org taken down by US DNS provider. URL `http://news.netcraft.com/archives/2010/12/03/wikileaks-org-taken-down-by-us-dns-provider.html`.

[2] Wikipedia. Zooko's triangle, . URL `http://en.wikipedia.org/wiki/Zooko%27s_triangle`.

[3] Aaron Swartz. Squaring the triangle: Secure, decentralized, human-readable names. URL `http://www.aaronsw.com/weblog/squarezooko`.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. URL `https://bitcoin.org/bitcoin.pdf`.

[5] appamatto. Bitdns and generalizing bitcoin. URL `https://bitcointalk.org/index.php?topic=1790.0`.

[6] J Bonneau, J Clark, E Felten, J Kroll, A Miller, and A Narayanan. On decentralizing prediction markets and order books. In *Workshop on the*

*Economics of Information Security*, 2014. URL http://weis2014.econinfosec.org/papers/Clark-WEIS2014.pdf.

[7] Domain name specification, 2015. URL https://wiki.namecoin.info/index.php?title=Domain_Name_Specification.

[8] khalahan. Nmcontrol. URL https://github.com/khalahan/nmcontrol.

[9] Freespeechme website. URL http://www.freespeechme.org/.

[10] Greg Slepak. Dnschain. URL https://github.com/okTurtles/dnschain.

[11] Opennic project. URL http://www.opennicproject.org/.

[12] Keybase. Keybase. URL https://keybase.io/.

[13] Luke A Walker. Icann's uniform domain name dispute resolution policy. *Berk. Tech. LJ*, 15:289, 2000.

[14] William L Silber. Marketmaker behavior in an auction market: an analysis of scalpers in futures markets. *The Journal of Finance*, 39(4):937–953, 1984.

[15] John R Ottensmann. Urban sprawl, land values and the density of development. *Land economics*, pages 389–400, 1977.

[16] Atila Abdulkadiroglu and Tayfun Sönmez. Matching markets: Theory and practice. In *Advances in Economics and Econometrics (Tenth World Congress)*, pages 3–47, 2013.

[17] Daniel. Proposal: Domains should be regularly auctioned. URL https://forum.namecoin.info/viewtopic.php?f=11&t=2003.

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[19] Juan Pablo Timpanaro, Chrisment Isabelle, and Festor Olivier. Monitoring the i2p network. 2011. URL https://hal.inria.fr/hal-00653136/document.

[20] ANTPY Github. Bitcoin github commit history. URL https://github.com/bitcoin/bitcoin/commit/5253d1ab77fab1995ede03fb934ed.

[21] Ethereum white paper, 2015. URL https://github.com/ethereum/wiki/wiki/White-Paper.

[22] Bitcoin Wiki. Agents. URL https://en.bitcoin.it/wiki/Agents.

[23] Wikipedia. Domain name front running, . URL https://en.wikipedia.org/wiki/Domain_name_front_running.

[24] phelixbtc. Bitcoin github. URL https://github.com/phelixbtc/antpy.

[25] Namecoin Forum. Idea for better fee structure. URL https://forum.namecoin.info/viewtopic.php?p=6653&sid=2cf37d245bbc611a8d93ad6ac23b9982.

[26] .p2p (bitshares dns), 2015. URL http://wiki.bitshares.org/index.php/.p2p_%28BitShares_DNS%29.

[27] Emercoin, 2015. URL http://emercoin.com/.

[28] Mastercoin protocol specification, 2015. URL https://github.com/mastercoin-MSC/spec.

[29] Meni Rosenfeld. Overview of colored coins, 2012. URL https://bitcoil.co.il/BitcoinX.pdf.

[30] Jeremy Clark and Aleksander Essex. Commitcoin: Carbon dating commitments with bitcoin. In *Financial Cryptography and Data Security*, pages 390–398. Springer, 2012. doi:10.1007/978-3-642-32946-3_28.

[31] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 443–458. IEEE, 2014. doi:10.1109/SP.2014.35.

[32] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology–CRYPTO 2014*, pages 421–439. Springer, 2014. doi:10.1007/978-3-662-44381-1_24.

[33] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41. ACM, 2014. doi:10.1145/2660267.2660380.

[34] Vitalik Buterin. Daos, dacs, das and more: An incomplete terminology guide, 2014. URL https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/.

[35] Bing blogs: Making search yours, 2011. URL http://blogs.bing.com/search/2011/02/10/making-search-yours/.

# A List of significant .bit domains

Below is a list of names counted towards the last entry in Table 2. These domains were determined by the proccess described in Section 4.

- alt-freedom.bit
- bangkokgroup.bit
- bitcoinpl.bit
- bitcoinquotes.bit
- carnicominstitutemirror.bit
- changepurse.bit
- columbo.bit
- darkfur93.bit
- dcinvestments.bit
- dealing.bit
- deathrowdemocracy.bit
- dot-bit.bit
- dotbitkittypix.bit
- feens.bit
- hewgill.bit
- hosting.bit
- kk-cabrio.bit
- lapan.bit
- medicinalgenomics.bit
- megabrutal.bit
- michail.bit
- mikeward.bit
- namecoin-test-suite-1.bit
- onemillionpixels.bit
- peterboswell.bit
- rmgx.bit
- tuler.bit
- wikimouto.bit