# Distributed Decentralized Domain Name Service

Brendan Benshoof Andrew Rosen
Anu G. Bourgeois Robert W. Harrison

Georgia State University

May 27th, 2016

# What is D$^3$NS

Distributed Decentralized Domain Name Service

- The goal is to create a secure distributed DNS.
- Requirements
  - Decentralization
  - Authentication
  - No end user modification

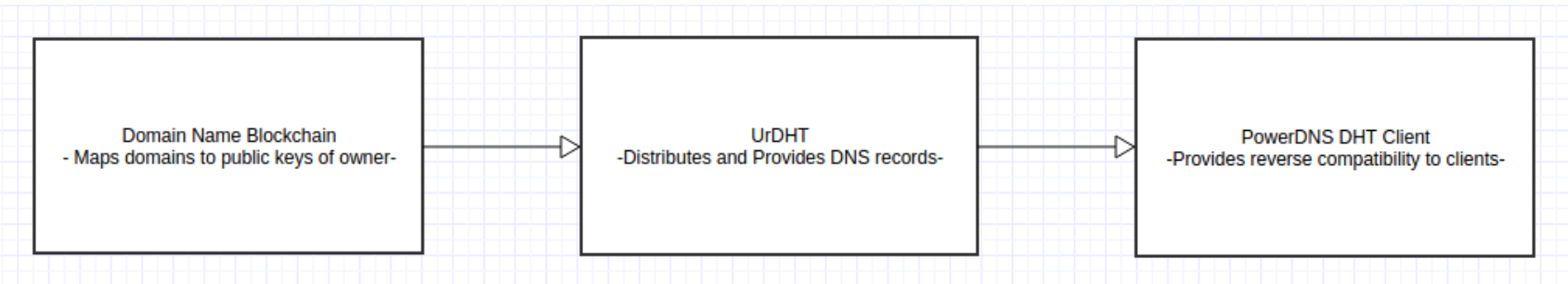
# Motivation

- Recent events have demonstrated that centralized authorities are not as secure a previously hoped.
  - There is little cryptographic protection against the subpoena.
  - Poorly constructed laws targeting DNS.
- A distributed approach for authentication is much less vulnerable.

# Components

- Domain Name Blockchain.
- Distributed Hash Table (UrDHT).
- DNS server frontend (PowerDNS)

# Architecture



# Distributed Hash Tables

- Means of organizing communication and responsibility in a P2P network
- Each peer is responsible for a verifiable span of hash values
- Facilitates one-to-one communication and one-to-many communication

# UrDHT

- Abstract DHT backend written in python.
- Handles:
  - Arbitrary DHT.
  - Plugin Services
- Subject of other research

# UrDHT Details

- DHT organization mechanism.
- Uses Voronoi regions on an $n$-dimensional torus to assign responsibility.
- Can define how to compute the regions to emulate almost any DHT topology.
- Node responsibility:
  - Node is responsible for its space, defined by its neighbors.
  - If a node leaves/fails, each neighbors assumes that it is responsible until corrected by maintenance.

# Fault Tolerance

- Churn creates a period where i/o can fail. With UrDHT:
- Reads of backed up data are successful.
- Writes to the region are successful.
- Reads of **new** data are vulnerable until it is backed up ($< 2$ sec currently).
- This means a much smaller window. Writes never fail.[1]

[1] They may occur out of order

# Cool Thing UrDHT Can Do

- Embed problem spaces into DHT topology
- Minimal latency based routing
- Basically turns routing into best-search first.

# DNS Blockchain

- Using a technique similar to bitcoin, we can assign domain names as reward for mining new blocks and transfer domains between owners.
- An ‘owner’ in this context is a public key
- These public keys can be used to verify stored DNS records by their signature records.

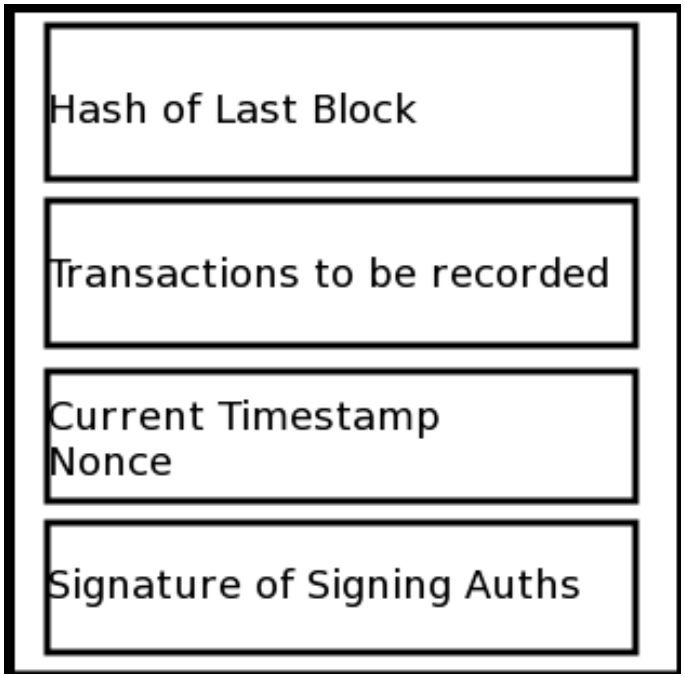



Figure: Contents of an individual block.

# PowerDNS

- Well established authoritative DNS server software.
- Provides easy interface for custom applications.

# Current DNS

- ICANN is the final arbiter on who owns what domain
- ICANN maintains and organizes the TLD authoritative name servers
- Third party verifiers act to authenticate DNS records

# P2P-Based DNS

- The shared block chain is the final arbiter of who owns what
- The DHT organizes and maintains the authoritative TLD servers
- The block chain acts to authenticate DNS records

# Man in the Middle In a DHT

- Need to have a distributed, reliable way to authenticate
- Given: an existing network where nodes have exchanged keys securely
- Given: a new peer who wishes to join the network and share their public key

# Prevention

- At least 2 members of the network interrogate the new peer for its public key
- Those interrogators compare their results
- If those results match:
  - The new peer creates an authentication record
  - The interrogators sign that record
  - The new record is distributed across the network
- If the results do not match
  - An attack is detected and reported to the new peer by all authenticating servers.
  - A member of the network may make a ban of the compromised peer
  - Otherwise the joining process can be repeated.

# Conclusions

- Proof of concept of a decentralized and distributed top-level DNS.
- Fully reverse compatible.
- Offers greater security.
- Any organization can create their own secure verification server.

# How Does This Differ From Namecoin?

- Transparent to end users.
- Namecoin doesn't scale efficiently, our system.