

Fault-tolerant Root

High Level Technical Architecture

<https://github.com/songlinjian/distributed-root>

Davey Song

@Montreal Yeti Meeting 23 July 2019

Goal and Motivation

- To design and implement a Root System in Yeti with NO centralized role (or as less as possible)
 - Decentralization with properties of fault-tolerance and resilience
 - End of physical meetings with Root System Automation
 - Broaden participation
- It results from ongoing discussions during the launch of phase-2 for Yeti DNS Project towards a decentralized Root system.

Backgrounds

- The idea of sharing control over the root and a "masterless" approach --ITI2014
- A mechanism called threshold validation to improved trust and redundancy for DNSSEC Keys --draft-ihren- dnsext-threshold-validation
- To remove the technical requirement for a central authority over the root for next-generation DNS" --RFC8324
- Yeti Testbed implemented 3 DNSSEC Signer of root -- RFC8483
- Threshold Crypto signatures in root signing --NIC Chile's work
 - Threshold Cryptography C Library (Victor Shoup) for RSA
 - Implement a PKCS#11 provider to fit current DNSSEC context

Root function: Editorial, Signing and Publication

- Editorial function (what names should exist, who are delegees?)
 - IETF, IANA, ICANN
- Signing function (what keys will be used, by whom, with what policy?)
 - IANA/ICANN, Verisign, Yeti
- Publication function (what servers, operated by whom?)
 - Rootops, RFC 7706, Yeti

Root function: Editorial, Signing and Publication

- Editorial function (what names should exist, who are delegees?)
 - IETF, IANA, ICANN
- Signing function (what keys will be used, by whom, with what policy?)
 - IANA/ICANN, Verisign, Yeti
- Publication function (what servers, operated by whom?)
 - Rootops, RFC 7706, Yeti



Targeted Area

A Decentralized Framework of Yeti Root

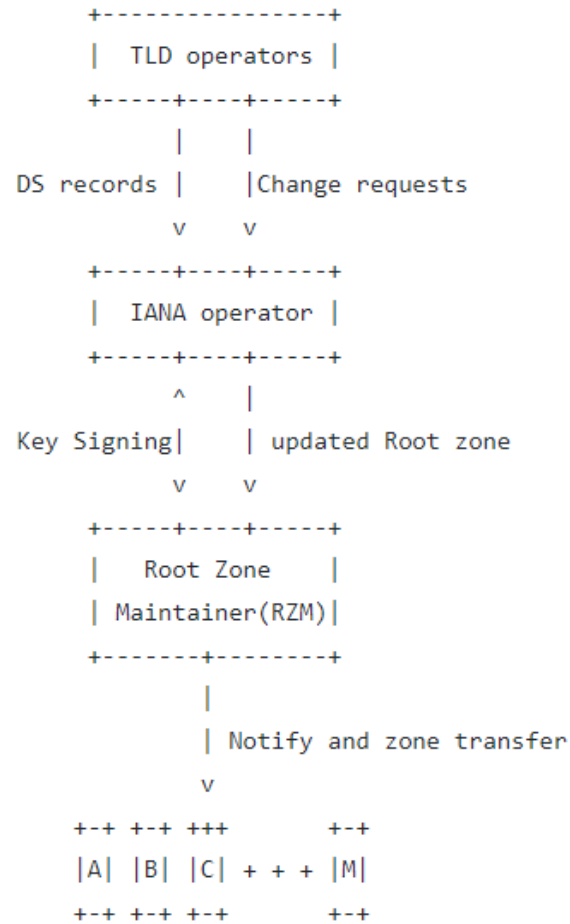


Figure 1: Root zone update process and data flow

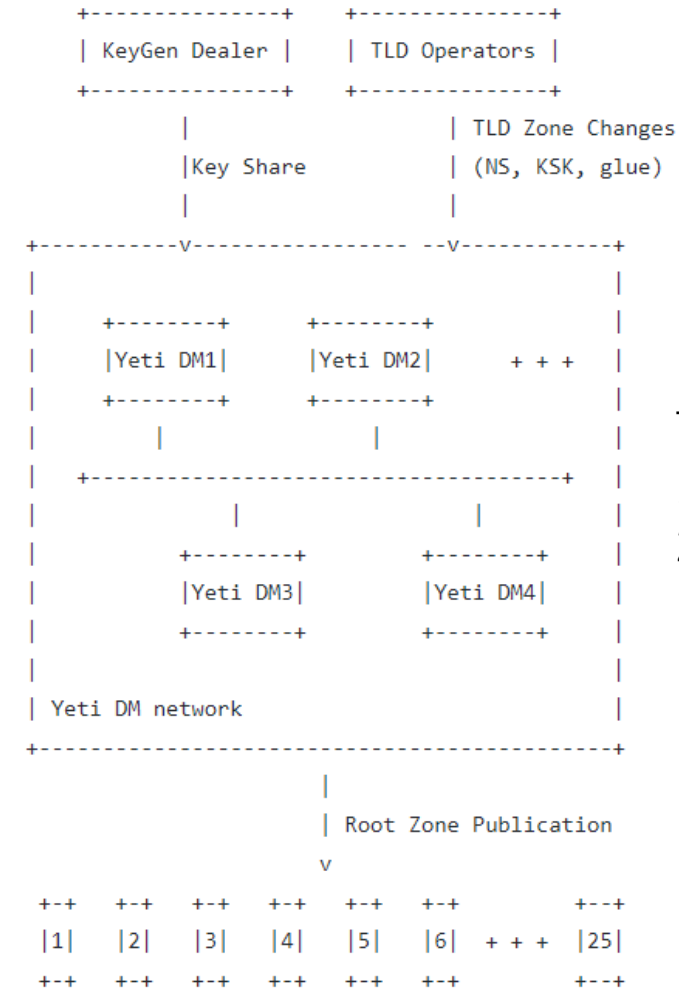
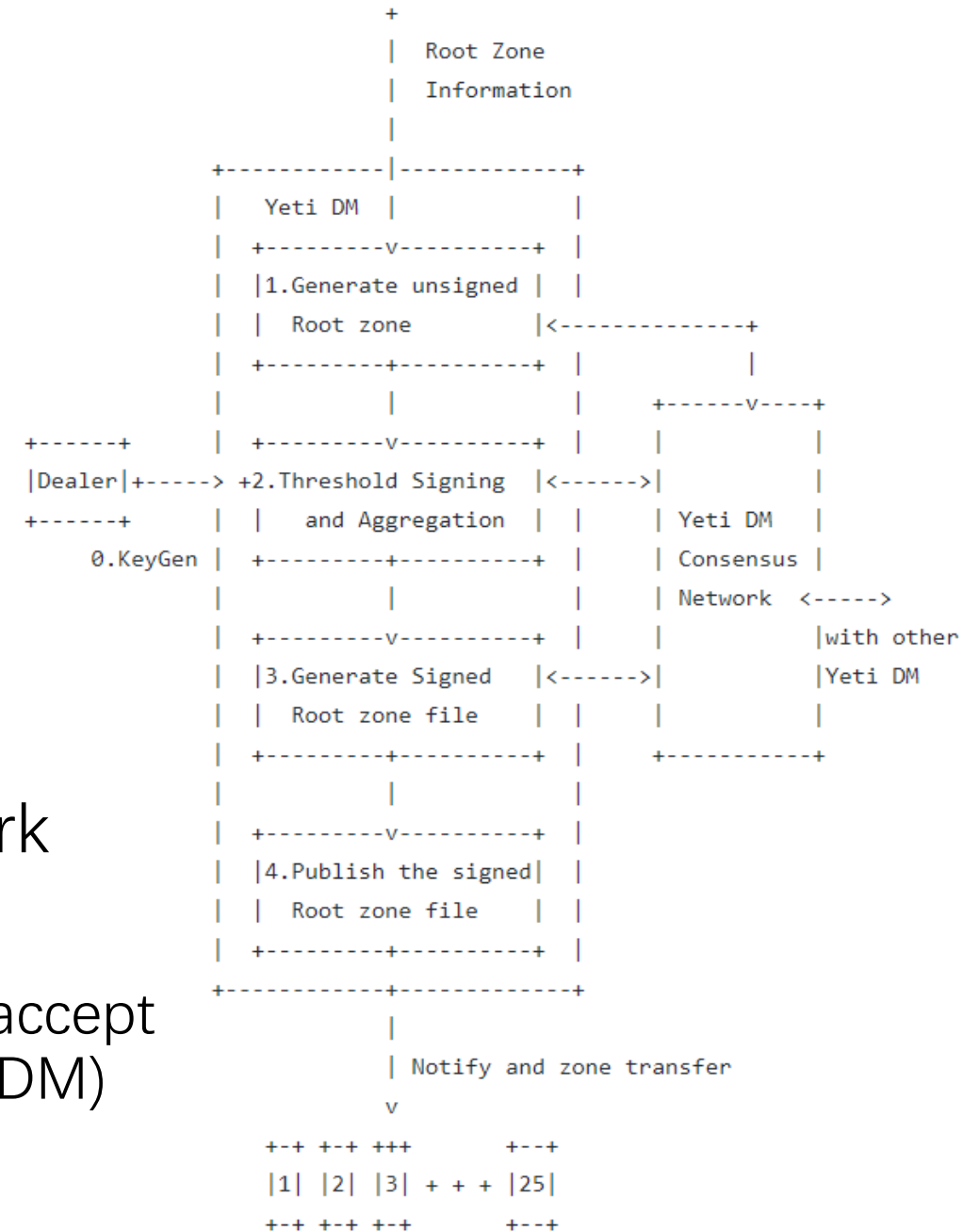


Figure 2: The framework in Fault-tolerant Root

- Two key components
1. Threshold signing
 2. Consensus algorithm

The Data Flow and Process

- Input : Root zone information (pull)
- Yeti DM procedures
 - 1. Generate unsigned Root zone
 - 2. Threshold Signing and Aggregation
 - 3. Generate Signed Root zone file
 - 4. Publish the signed Root zone file
- Yeti DM connected to Consensus Network
- One change on Root slave
 - To verify if it is a valid root zone file before accept a root zone (in case of a compromised Yeti DM)



Future work

- Complete the high-level document of Fault-tolerant Root
- Prototype and test of a threshold cryptography algorithm (Appendix A)
 - NIC Chile's implementation of Victor Shoup's paper only for RSA
<https://github.com/niclabs/tchsm-libtc>
 - Implement threshold BLS signatures with Paring-based Crypto (PBC)(RSA and ECDSA) <https://github.com/asonnino/bls>
- Prototype and test of a consensus algorithm
 - Raft-like algorithm (Appendix B)
 - Consensus network using Blockchian (or distributed ledger)
- Enhance the software and stress tests
- Operational testbed in Yeti
- Technical Deliverables to describe this system and our experience