Project presentation CS-438 EPFL
Fall Semester 2020-2021

Hippocrates:
Decentralized, Secure File Storage.

Team Members

Francesco Intoci

Utku Görkem Ertürk

Aaron Lippeveldts

Workload:

- Francesco : DKG, Cothority, Doctor and Patient APIs (File encryption / decryption).
- Görkem : Blockchain & Paxos, Basic structure of DHT with linear search.
- Aaron : Blockchain & Paxos, DHT with finger tables and reliability, CLI

What is Hippocrates?

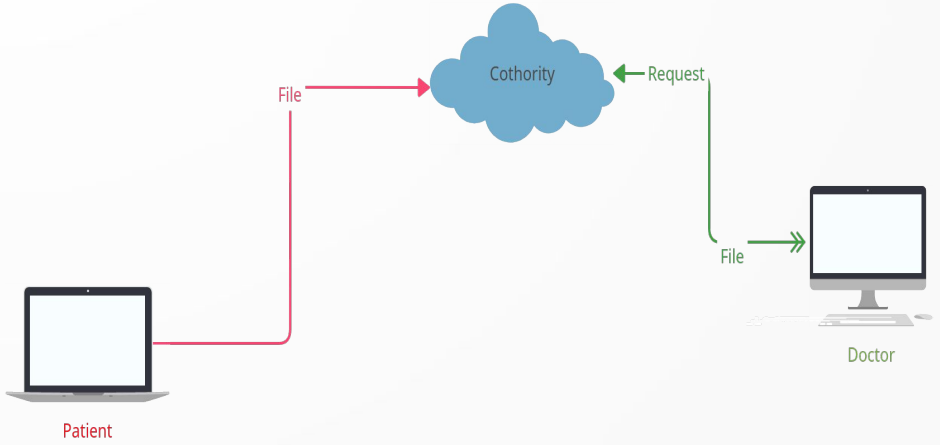
- Hippocrates is a decentralized system which allows users to upload files they want to share with authenticated users in a **secure, reliable** way.
- **Real World Use Case**: patients upload their medical records which can be consulted only by trusted doctors.
- Build in: Reliability, Privacy and Distributed Storage

A patient wants to share a file...

- First, he **splits** the file into chunks and he **encrypts** them using AES-CBC-192. He **distributes** the encrypted copies.
- He sends **encrypted meta-data** and the **(encrypted) key** in a secure way to a trusted, collective, authority: the **Cothority**.

A doctor wants to retrieve a file...

- First, he will **sign** a request for a specific file using HMAC-SHA256.
- If granted, the Cothority sends the meta-data to the doctor, who will fetch chunks to reassemble the file and decrypt it.



So far so good...right?

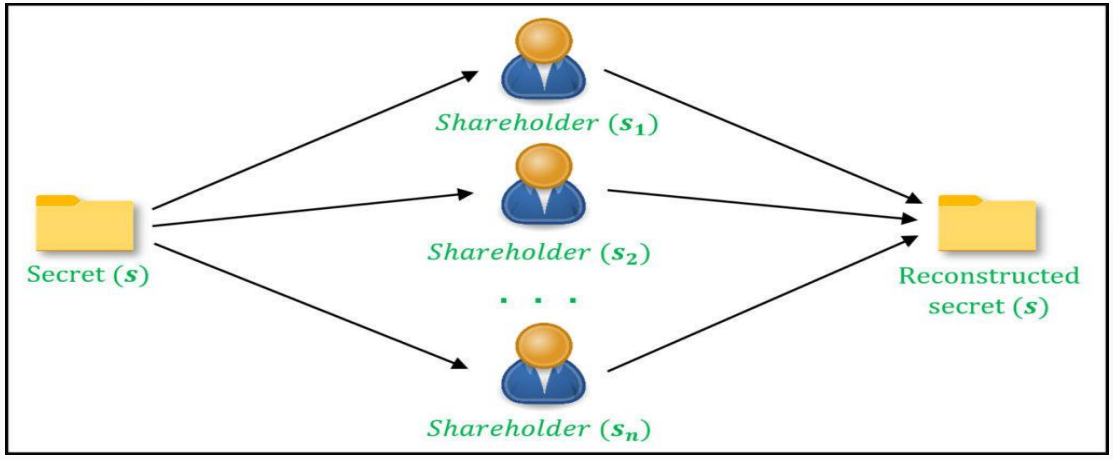
- **Problem:** for privacy reason we do not want single Cothority node to be able to decrypt the meta-data of a file.
- **Problem:** how can doctors decrypt meta-data? Sending Cothority private key would be disastrous for security!

Solution: In Crypto we trust

- In order to overcome these problems, Cothority will use a DKG protocol: **Distributed Key Generation Protocol**.
- DKG allows Cothority nodes to generate a secret key, public key pair: **each node will know the public key, but just part of the secret key**.
- To reassemble the secret key, ***t out of n*** Cothority nodes will be needed.

Secret's Sharing and VSS:

- DKG relies on two cryptographic primitives: **Secret Sharing and Verifiable Secret Sharing**.
- VSS: Dealer will pick a secret. Each Participant will receive a **share of the secret and a verifiable commitment**.
- Still a problem: the Dealer **does know** the secret.



Distributed Key Generation

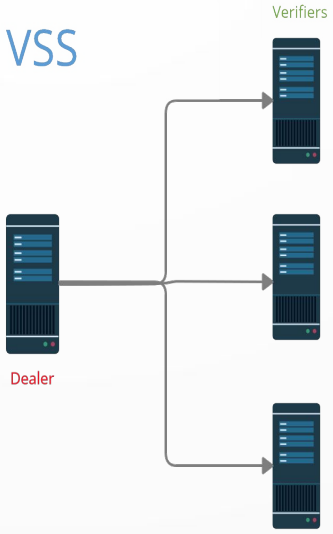
- DKG runs **several instances of VSS in parallel**.
- Several steps:

  1 - **exchange public keys** (used for verify signature of Deals).

  2- send **Deals**, consisting of that participant **share** of the secret and a public **commitment**.

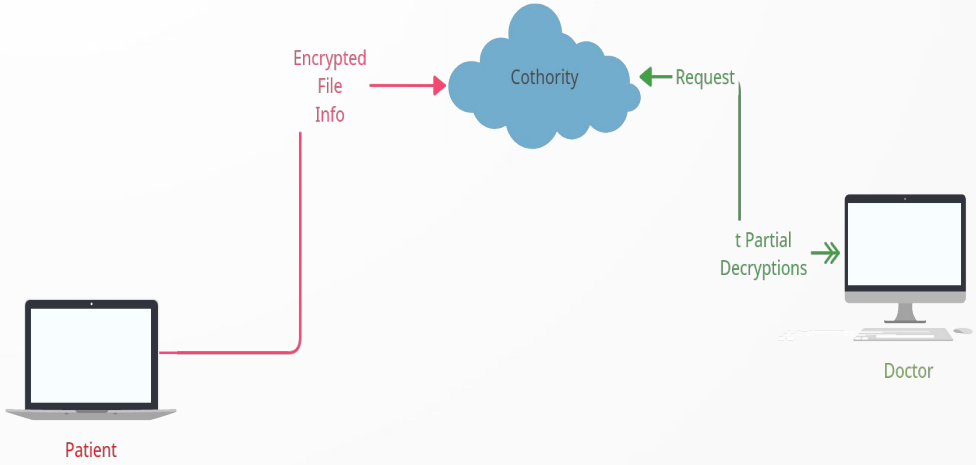
  3 - reply to a Deal with a **Response**, which will be broadcasted among all the participants.

- In the end a subset of the Cothority nodes will be considered ***qualified***, based on the Deals and the Responses collected.



Sending Partial Decryptions

- Still a problem: how can doctors decrypt?
- Solution: send doctors **partial decryption** of meta-data (each node of the Cothority will partially decrypt with its share).
- After collecting **t partial decryption**, doctors will recover the encryption key and decrypt. Note that in this way doctors **will not recover the secret key**.



DKG in practice:

- Our solution heavily relies on the **Kyber** library, an advanced cryptographic library developed by the DEDIS lab at EPFL.
- Many assumptions were made:

  1 - Cothority nodes are part of a **fully connected network.**

  2 - **Addresses of Cothority Nodes** are **known.**

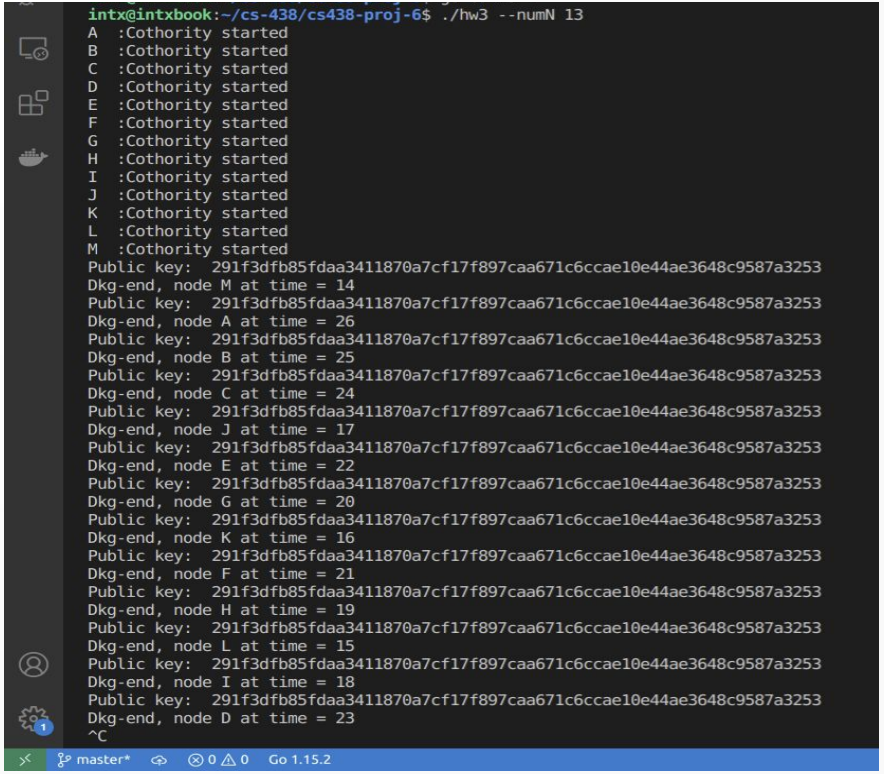
  3 - Cothority Nodes follow the protocol (**no invalid Deals**. **Communication faults** accepted.)
- Our biggest **challenge** was to **adapt the *dkg* packet** so that it could work in a realistic scenario such as Hippocrates...

The curse of UDP...

- The amount of network traffic during DKG protocol, not taking in account retransmissions, is roughly ~ **$O(n^2)$**.
- Our first implementation used **unreliable UDP broadcast** for Public Keys and Responses to Deals. 3 nodes with large timeouts couldn’t complete DKG protocol due to packet losses.

...and the bless of Gossip

- We decided to wrap Public Key Messages and Response Messages into Rumor messages, taking advantage of the **Rumor Mongering and Anti-Entropy** logic we have developed in Peerster.
- Gossip Protocol proved to be highly reliable, as we managed to complete the DKG protocol with shorter timeouts and even with more nodes.



Comments:

- Analytically, we have found an exponential function interpolating our test data. We have decided to select an **exponential curve** to fit our data since the retransmissions due to **Gossiping protocol increasing the quadratic complexity**.

- We assumed that Cothority nodes are **not synchronized**. In order to obtain convergence to the **same subset of Qualified** nodes, we had to set a large enough timeout.

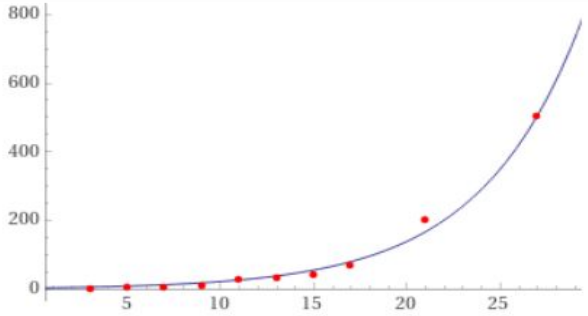
Input interpretation:

fit	data	{{3, 1}, {5, 2}, {7, 3}, {9, 7}, {11, 27}, {13, 30}, {15, 38}, {17, 69}, {21, 200}, {27, 500}}
	model	exponential

Least-squares best fit:

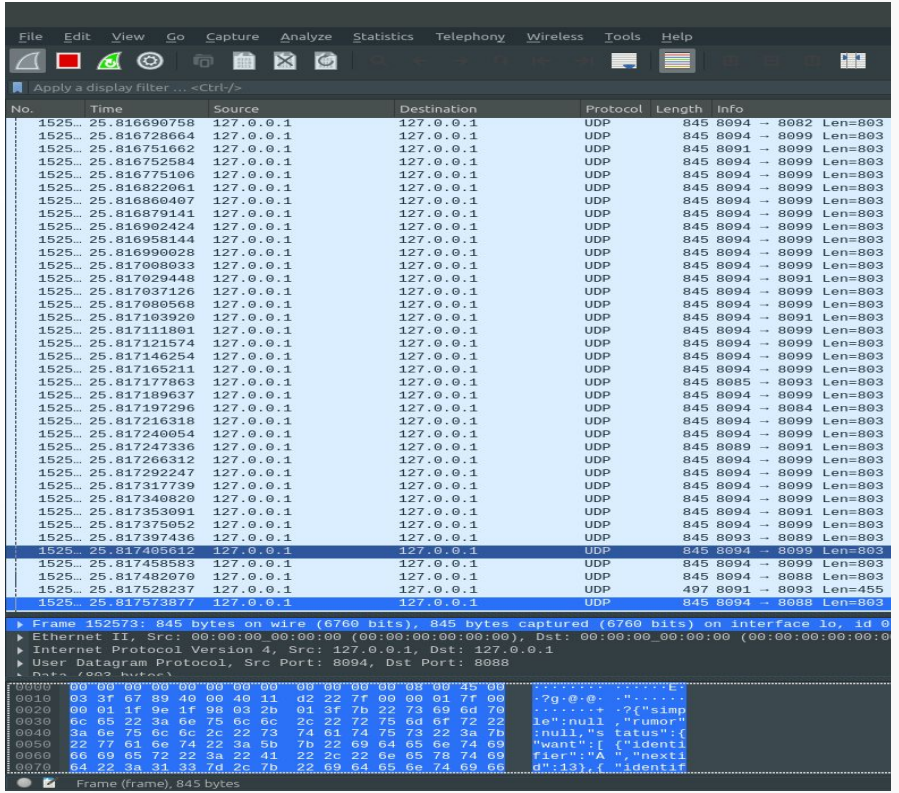
$3.48538\, e^{0.184349\, x}$

Plot of the least-squares fit:



Comments - II:

- The resulting plot suggests that the our DKG protocol exploiting Gossip protocol works well with **small-medium size networks**.
- **TTC increases exponentially** with the number of nodes.
- In conclusion, our solution can be applied only with a limited number of nodes, which could be a **realistic assumption** taking in account that the protocol is performed by a small set of trusted nodes.



Francesco Intoci

Blockchain & Paxos

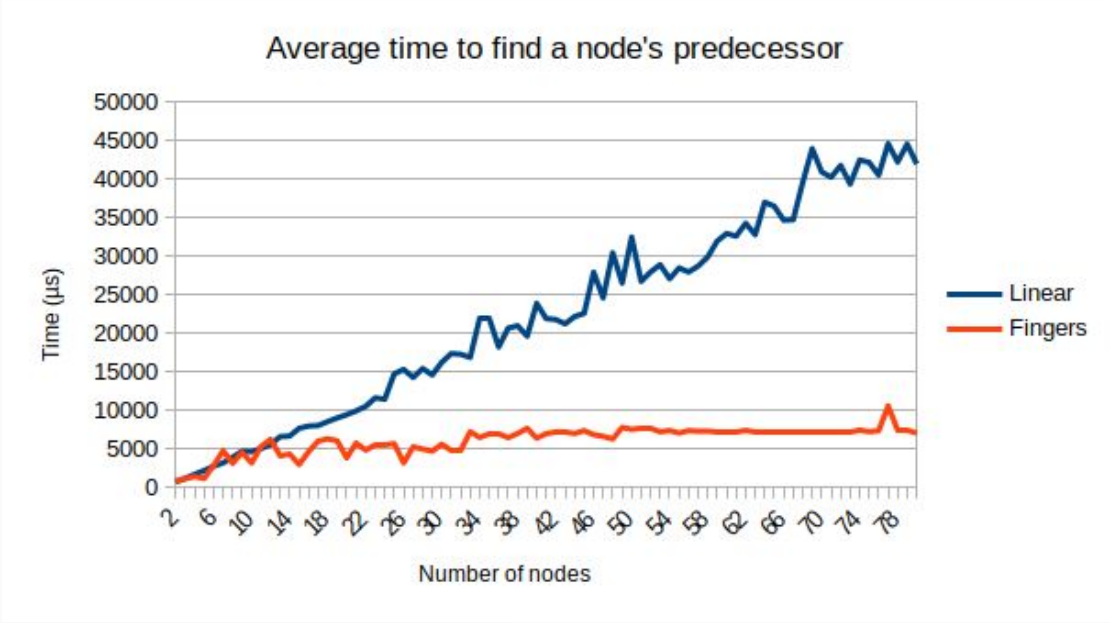
- We use **Blockchain** and **Paxos** to record file requests.
- Whenever a doctor requests a file, his name is immediately logged to the blockchain including a timestamp and IP address.
- **Auditability**, authorities can track who is culpable in case of data leaks.
- Because of the blockchain, malicious Cothority nodes **cannot alter records**.

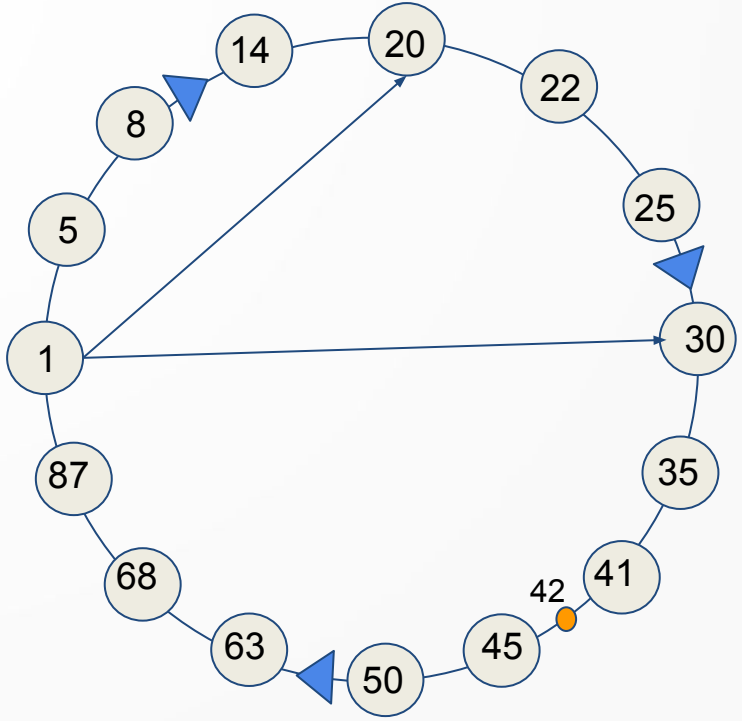
Utku Görkem Ertürk & Aaron Lippeveldts

Distributed Hash Table

- File chunks are stored in a DHT.
- All nodes participate in the DHT.
- Chord with **finger tables** and **redundant successors.**
- File chunks are stored **encrypted**, files names hidden using **hashing.**
- **Redundant copies** for every chunk in case a node goes down.

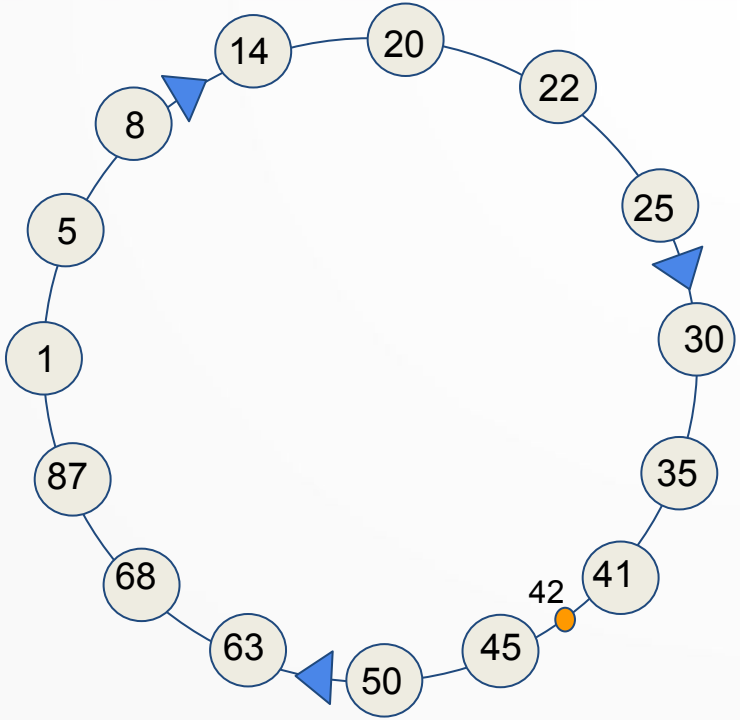
DHT: Linear vs Fingers



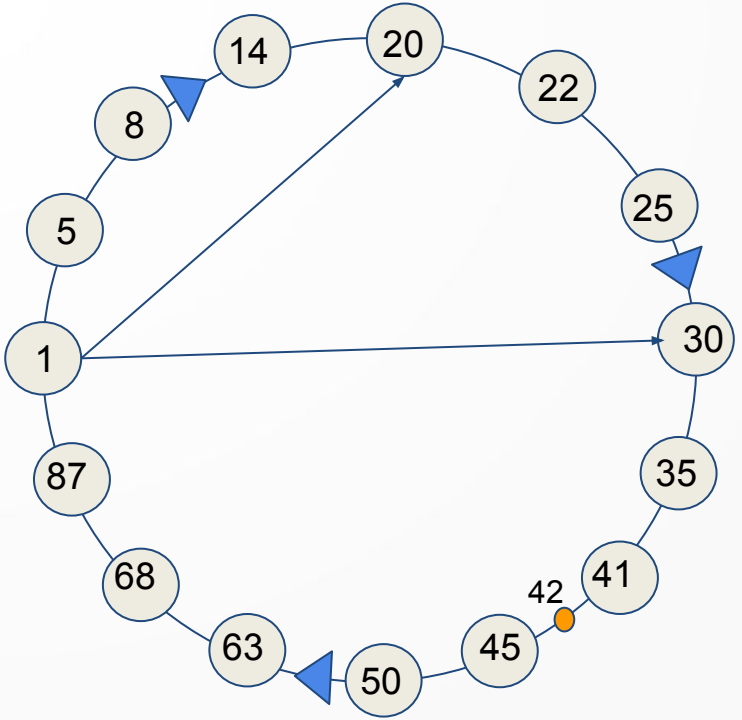


Utku Görkem Ertürk & Aaron Lippeveldts

DHT: Linear vs Fingers - I



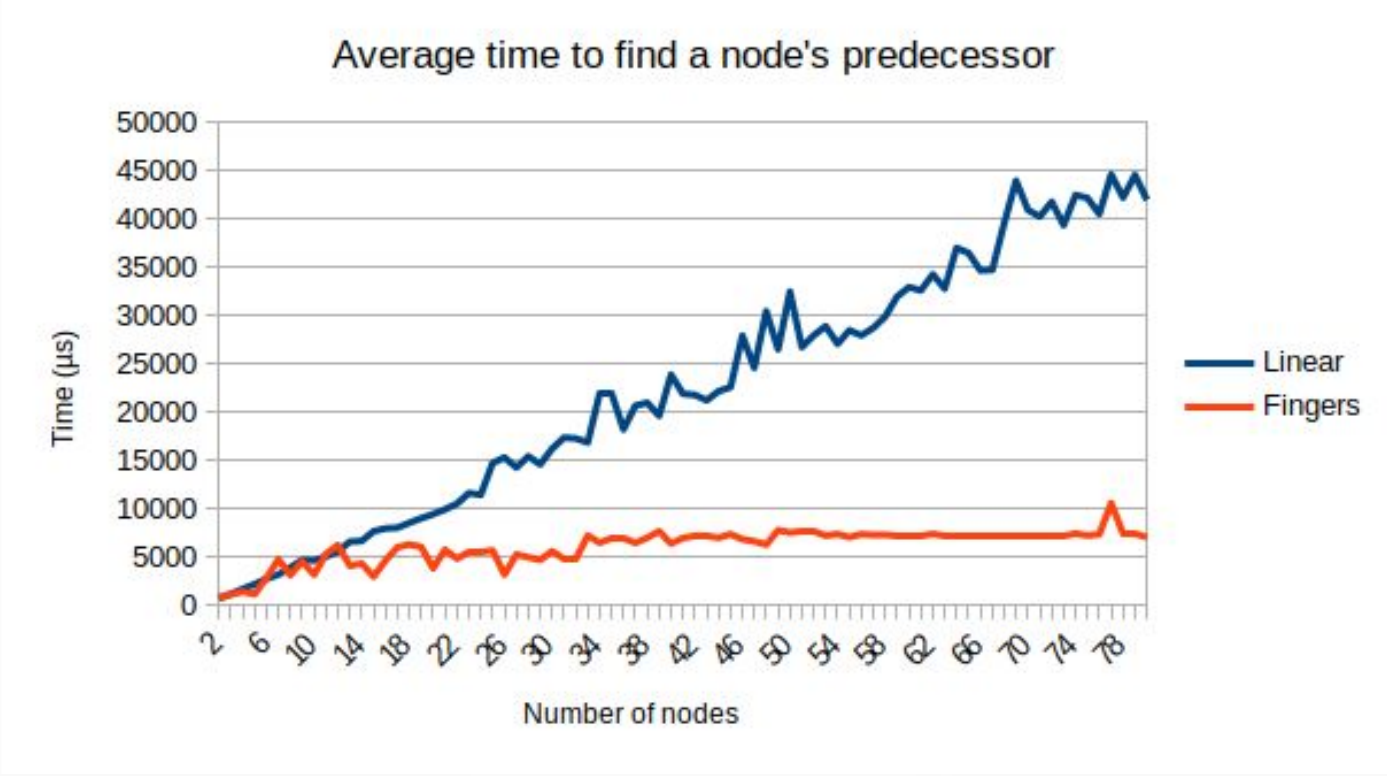
Linear



Fingers

Utku Görkem Ertürk & Aaron Lippeveldts

DHT: Linear vs Fingers - II



Linear vs 3 Fingers

Utku Görkem Ertürk & Aaron Lippeveldts

Overall Architecture