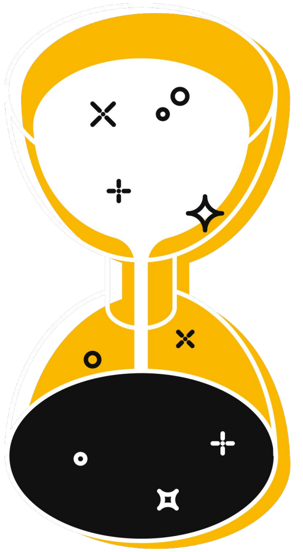
Roughtime:
Securing time for IoT devices

Marcus Dansarie, Netnod

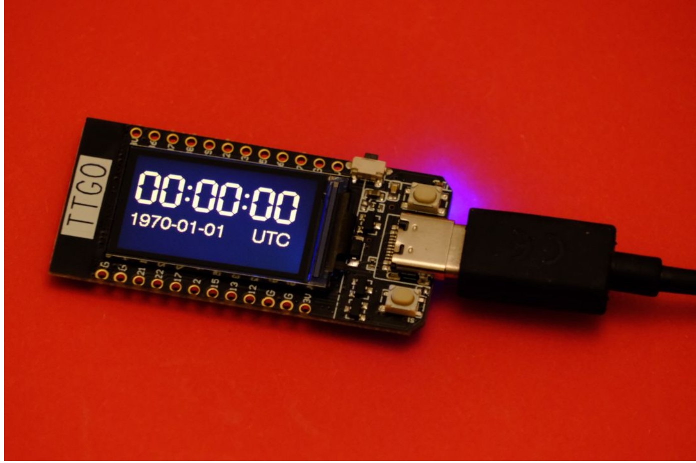
Why accurate time is important

- Many security critical protocols need accurate time
    - DNSSEC, secure domain name lookups
    - TLS, the basis of many other protocols
        - HTTPS, everything on the web
        - SMTPS, IMAPS, POP3S, secure mail
    - Accuracy requirement: within a few minutes or hours
    - Risks of not having accurate time
        - Fall back to insecure algorithms
        - Use old (maybe leaked) information
- The application itself might need time
    - Example: electronic door lock
    - May need more accurate time than minutes or hours

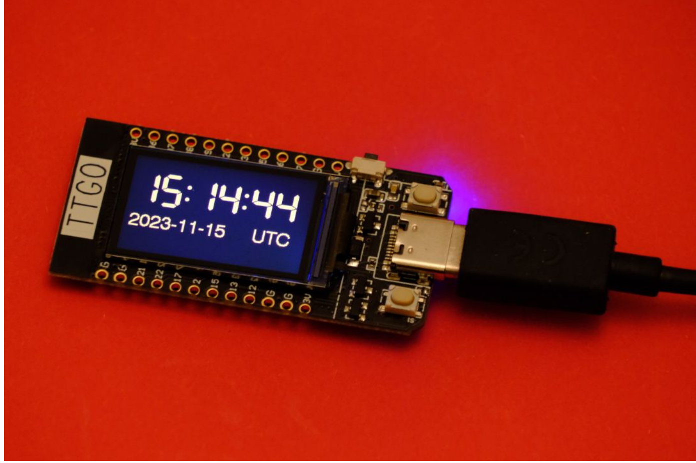


Keeping time

- All devices can keep time
  - When powered on
- But not when powered off
  - IoT devices may not have a Real Time Clock (RTC)
  - Raspberry Pi – has RTC hardware, but no battery backup by default
  - "Shipping mode"
    - Even with a battery the clock will not run before first power on because the battery is not connected
- "Ten year on the shelf problem"
  - A device can sit on a shelf for a long time before being turned on



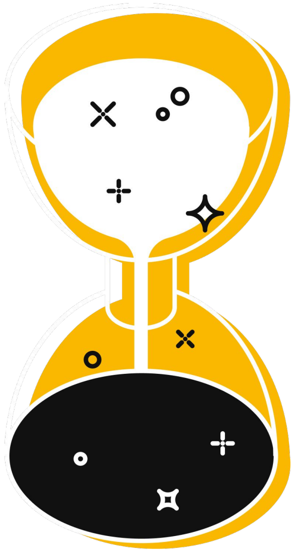
Getting time over the network



- NTP (Network Time Protocol)
    - Lacks scalable security
- NTS (Network Time Security)
    - Adds security to NTP
    - Bootstrapping problem
        - NTS depends on TLS
        - Which depends on having accurate time
    - Heavyweight, not suited for resource constrained devices
- Others (e.g. HTTP/HTTPS date header)
    - No security, or depends on TLS and thus has the bootstrapping problem
- What if a time server fails or is compromised?
    - A common configuration for NTP is to use only one server
    - Single point of failure

Possible solution: Roughtime

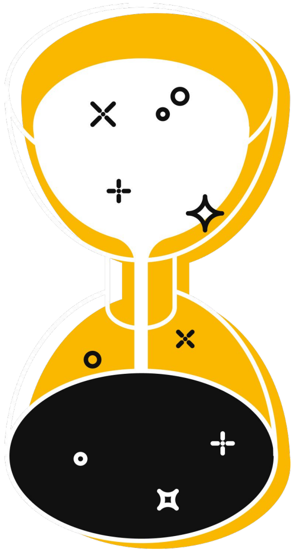
- Protocol is an IETF Draft
  - W. Ladd, M. Dansarie
- Started out as a way to verify system time
  - Secure
  - Not intended to replace NTP
  - Fairly low CPU usage and small memory footprint
- Netnod received RIPE community funding to help kickstart the development of Roughtime and the IETF draft.



Roughtime: concepts

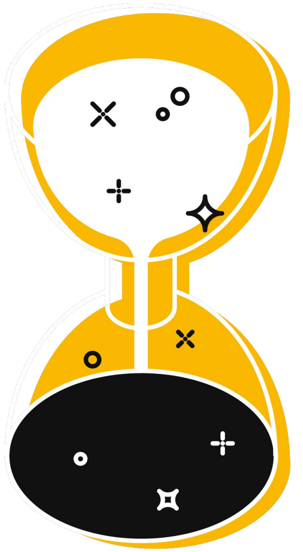
- Uses Ed25519 signatures, Merkle tree
- Hardcoded long-term public keys
  - Reduces bootstrapping problem
  - This is a tradeoff which turns it into a key distribution problem
- Client asks many servers for time
  - Requires consistency
  - Removes single point of failure/attack
- Intended for devices where the server list can be updated
  - Or part of a firmware update

- These concepts could be used with other time protocols



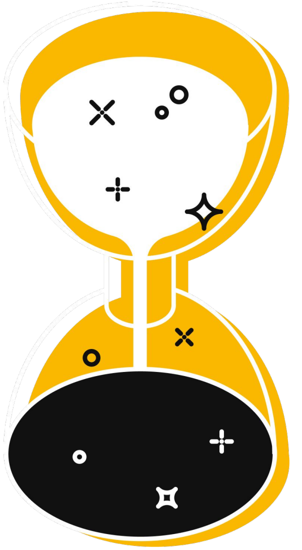
Roughtime: details

- A 32 byte nonce from the client is signed together with the timestamp
  - This is necessary for security anyway and is basically free
  - Allows signing of any document with a timestamp
  - A document can be the signed timestamp from another Roughtime server
  - This can provide cryptographic proof of misbehaving roughtime servers
  - Allows for accountability / auditing of roughtime servers
- Merkle tree reduces CPU load on the server
  - Ed25519 signing is a costly operation
  - Merkle tree spreads cost over multiple requests



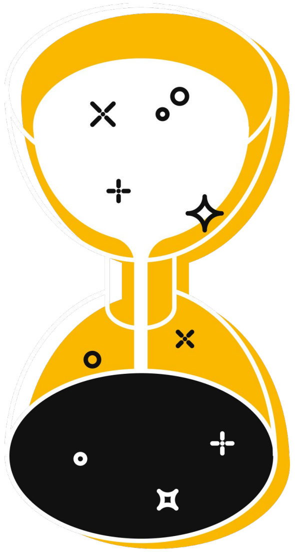
Roughtime: evolution

- It is now a decent generic time protocol
    - With significantly better accuracy than 10 seconds
        - Microsecond vs second resolution?
    - Which is secure (NTP is not)
    - Which can run on resource constrained clients (NTS is rather heavyweight)



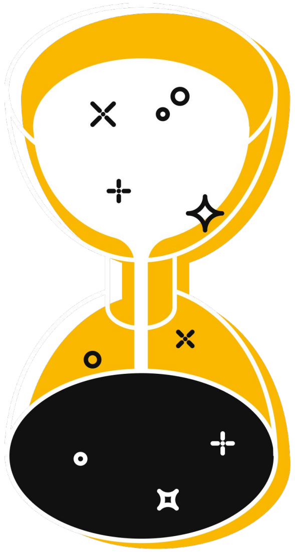
Next steps 1 – vendor requirements

- Reaching out to vendors to discuss their requirements
    - Use-cases for Roughtime
    - Application requirements
    - System requirements
    - Security considerations



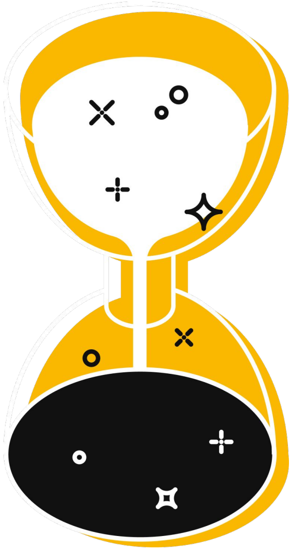
Next steps 2 – the draft

- Incorporate vendor requirements
- Refocus draft on main use cases and resolve complexity
- Reach consensus on core features of protocol
  - Timescale
  - Timestamp format and resolution
  - Protocol accuracy
  - Leap seconds and leap smearing
  - Protocol format



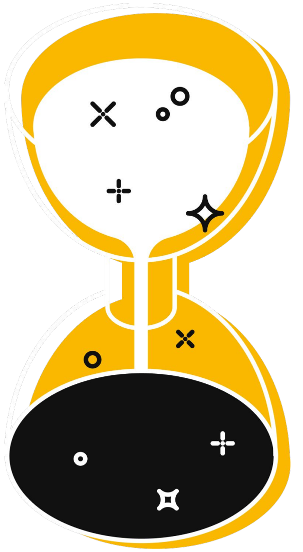
Next steps 3 – the implementations

- A number current implementations in C, Python and Go
- Support for different drafts - consolidation necessary
- Implementations specifically for resource constrained devices
- Interoperability of implementations
- Hackathon/interoperability testing later in 2024?



Next steps 4 – consensus and submit final draft

- Present at conferences
- Discuss on relevant mailing lists
- Establish working group consensus
- Submit Roughtime draft for publication



Resources

- Roughtime Draft
  - https://datatracker.ietf.org/doc/html/draft-ietf-ntp-roughtime
- Working client implementation of draft version 4, 5 and 7
  - https://vadarklockan.readthedocs.io
- Roughtime servers
  - Netnod: sth1.roughtime.netnod.se, sth2.roughtime.netnod.se (v7)
  - Marcus Dansarie: roughtime.se (v7)
- IETF NTP working group mailing list
- Mailing list: "proto-roughtime"
- Contact me: marcus@dansarie.se