AMERICAN UNIVERSITY OF BEIRUT

A Time-Locked Message Capsule Platform Based on Threshold Networks

Antoine Karam and Ghady Youssef

November 19, 2025

What is Time Lock Encryption?

- Timelock encryption (TLE) refers to encrypting a message or plaintext to the **future**.
- The encryptor of the message specifies an **unlock date** after which the ciphertext can be than automatically be decrypted.
- Before the unlock date, no one and even the encryptor will not be able to recover the plaintext.

Why is it useful?

- **Secure Electronic Voting:** Allow citizens to vote online while retaining the elections integrity and prevent fraud.
- **Sealed-bid Auctions:** Ensure fairness across all bidders by ensuring that bids are submitted anonymously and that nobody could change their bid.

Agent-based TLE

Trust you friend (or whoever you'd like) to release the key after the specified date.
Some issues with this approach:

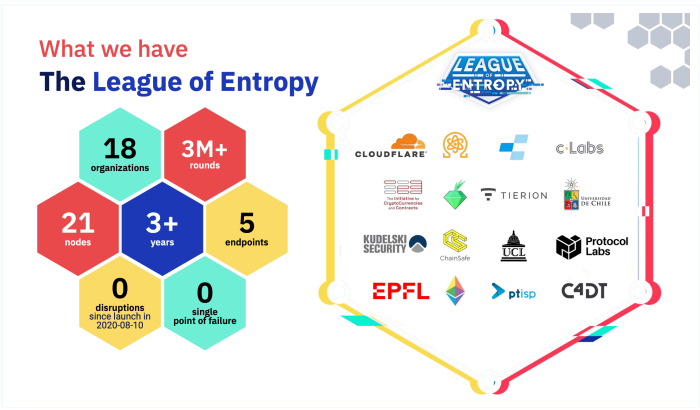
- Trust that our friend does not use the private key to peek at the message.
- Hope that our friend truely releases the keys are the designated date (not before nor later).
- They actually release the key.

Puzzle-based TLE

Similar to proof-of-work based systems. Do a lot of computations to solve a puzzle which will take “time”.
Some issues with this approach:

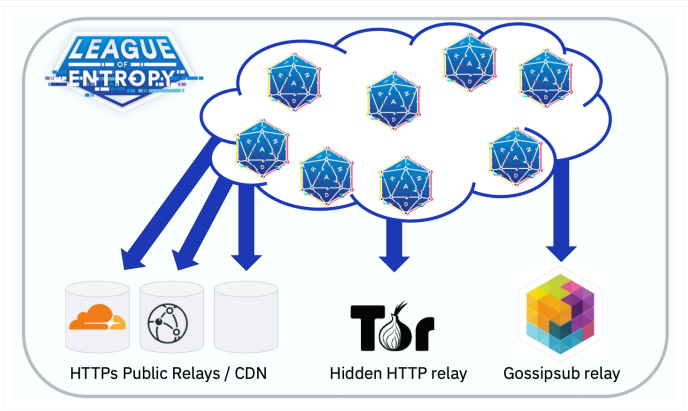
- Continuous hardrware and software advancements. Puzzle could become obsolete.
- High computational cost (energy, resources, compute power)

The League of Entropy



Source: Yolan Romailler

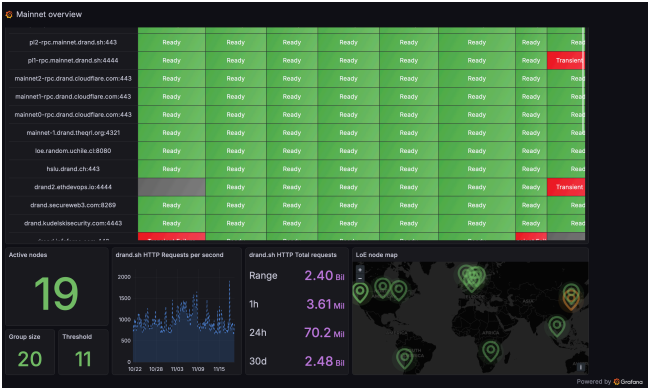
The League of Entropy



Source: Yolan Romailler

The League of Entropy

Some statistics



Source: Official Grafana dashboard hosted by drand

https://randamu.grafana.net/public-dashboards/d26c5131f5b04bb2b342ae15641278d6

Threshold Agent-based TLE

Rely on the threshold network (drand) to release the private keys at the designated time.

- Decentralized trust. No single entity has control over the network.
- Threshold security.
- Nobody can decrypt the messages until the network releases the key (and yes, even the decryptor)

Identity-based encryption

- The same as asymmetric encryption except that the public keys are the the identity of the recipient.
- What is an identity? Any string that represents the user: ID, e-mail address, phone number, etc.

Pairings

Pairing are maps that take a point $P \in \mathbb{G}_1$ and another $Q \in \mathbb{G}_2$ and return a point $X \in \mathbb{G}_T$. Bilinear maps have the following structure, by defining the pairing e:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \tag{1}$$

We have the following equalities that hold:

$$\begin{aligned} e(aG_1, bG_2) &= e(G_1, bG_2)^a \\ &= e(G_1, G_2)^{ab} \\ &= e(G_1, aG_2)^b \\ &= e(bG_1, aG_2) \end{aligned}$$

BLS signatures

Computing the signature

BLS signatures typically use bilinear pairings in order to sign and verify signatures.

- Private key is in $\mathbb{G}_2$, public key is in $\mathbb{G}_1$.
- Compute the private key $s \twoheadleftarrow \mathbb{G}_2$.
- Compute the public key $p = s \cdot G_1$.
- Compute the signature by hashing the plaintext pt (using a hash to curve function) multiplied by the public key. $\sigma = s \cdot H(pt)$.

BLS signatures

Verifying the signature

In order to verify the signature, we should verify that $e(G_1, \sigma) = e(p, H(m))$, we can validate this claim by the properties of the bilinear pairings.

$$
\begin{aligned}
e(p, H(m)) &= e(s \cdot G_1, H(pt)) \\
&= e(G_1, H(pt))^s \\
&= e(G_1, s \cdot H(pt)) \\
&= e(G_1, \sigma)
\end{aligned}
$$

Encryption

The Algorithm

Algorithm 1 Encryption algorithm. Source: The original timelock paper.

procedure Enc(pp, ρ, M)

Parse $pp \rightarrow (bg, P, H = (H_1, H_2, H_3, H_4))$

$PK_\rho \leftarrow e(P, H_1(\rho))$ ▷ round public key

$\sigma \leftarrow \{0,1\}^\ell$ ▷ nonce

$r \leftarrow H_3(\sigma, M)$

$U \leftarrow rG_1$ ▷ ephemeral public key

$V \leftarrow \sigma \oplus H_2((PK_\rho)^r)$ ▷ hiding commitment to nonce

$W \leftarrow M \oplus H_4(\sigma)$ ▷ one-time-pad

return $(ct = (U, V, W), \tau = r)$

end procedure

Encryption

Hybrid Encryption

Use the timelock encryption scheme to encrypt a 16 byte symmetric key that will be used to encrypt the file.

- Encrypt the file using `age-encryption`
- Use timelock encryption to encrypt (encapsulate) the 16 byte file key
- Output a file which contains:
  - Header: Encapsulated file key
  - Data: Encrypted binary data (file name + contents)

Decryption

Algorithm 2 Decryption algorithm. Source: The original timelock paper.

procedure Dec($pp, \rho, \pi_\rho, ct_\rho$)
  Parse $ct_\rho \rightarrow (U, V, W)$
  $\sigma' \leftarrow V \oplus H_2(e(U, \pi_\rho))$
  $M' \leftarrow W \oplus H_4(\sigma')$
  $r \leftarrow H_3(\sigma', M')$
  if $U = rG_1$ **then**
    return M'
  else
    return $\perp$
  end if
end procedure

Decryption

$$
\begin{aligned}
\sigma &= V \oplus H_2((e(U, \pi_p)) \\
&= \sigma \oplus H_2((PK_p)^r) \oplus H_2(e(U, \pi_p)) \\
&= \sigma \oplus H_2(e(P, H_1(p))^r) \oplus H_2(e(U, \pi_p)) \\
&= \sigma \oplus H_2(e(P, H_1(p))^r) \oplus H_2(e(rG_2, sH_1(p))) \\
&= \sigma \oplus H_2(e(sG_2, H_1(p))^r) \oplus H_2(e(rG_2, sH_1(p))) \\
&= \sigma \oplus H_2(e(G_2, H_1(p))^{rs}) \oplus H_2(e(G_2, H_1(p))^{rs}) \\
&= \sigma
\end{aligned}
$$

Software Used

- **Backend & API:** Implemented in Rust using `axum`
- **User Interface:** Web application implemented in React.js
- **Storage:** MinIO (S3-compatible) for file storage, SQLite for capsule metadata
- **File Encryption:** `age-encryption` for symmetric file encryption

Potential Threats

- **Compromised Server:**
  - Encryption happens at the client level.
  - A compromised server cannot access plaintext data directly.
- **Compromised drand network:** If an attacker manages to compromise the threshold of the network and takes control, they can potentially release malicious signatures and decrypt messages. Though this is practically infeasible!

Future Work

Post-Quantum Migration

- BLS signatures are still based on elliptic curves
- Broken in a post-quantum world
- We can use this architecture with post-quantum secure schemes

Future Work

Secure Electronic Voting

- **Commitment Schemes:** Enable users to commit to their vote securely and reveal them later.
- **Security issues:** Practical implementation at the national level is challenging due to security issues such as vote forgery.

Timelock encryption could be a step toward a future where citizens can vote online safely, without the risk of interference in the voting process!

Thank You!

Most of this work is based on:
Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. tlock: Practical timelock encryption from threshold BLS. Cryptology ePrint Archive, Paper 2023/189, 2023.
https://eprint.iacr.org/2023/189.pdf

Credits for the slides theme are attributed to Dr. Nadim Kobeissi