TIMELOCK ENCRYPTION

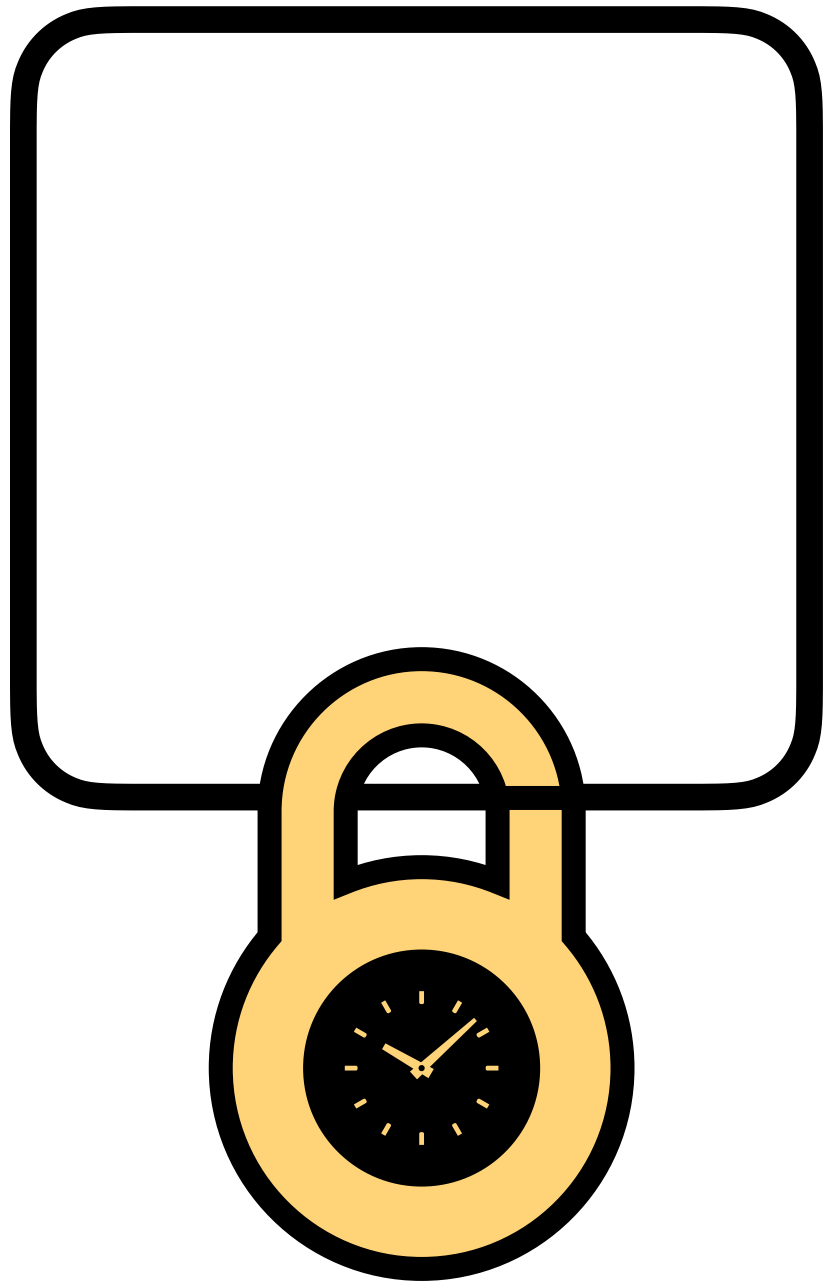
AN OVERVIEW AND RETROSPECTIVE

★ Kelsey Melissaris, Aarhus University
Yolan Romailler, Randamu

PART ONE OF TWO

Presented at NIST-STPPA7
January 16th, 2025

Timed-Release Crypto



- *To*: cypherpunks@toad.com
- *Subject*: Timed-Release Crypto
- *From*: tcmay@netcom.com (Timothy C. May)
- *Date*: Wed, 10 Feb 93 11:55:45 -0800

Cypherpunks,

I want to share with you folks some preliminary ideas on "timed-release cryptographic protocols," that is, methods for sending encrypted messages into the future.

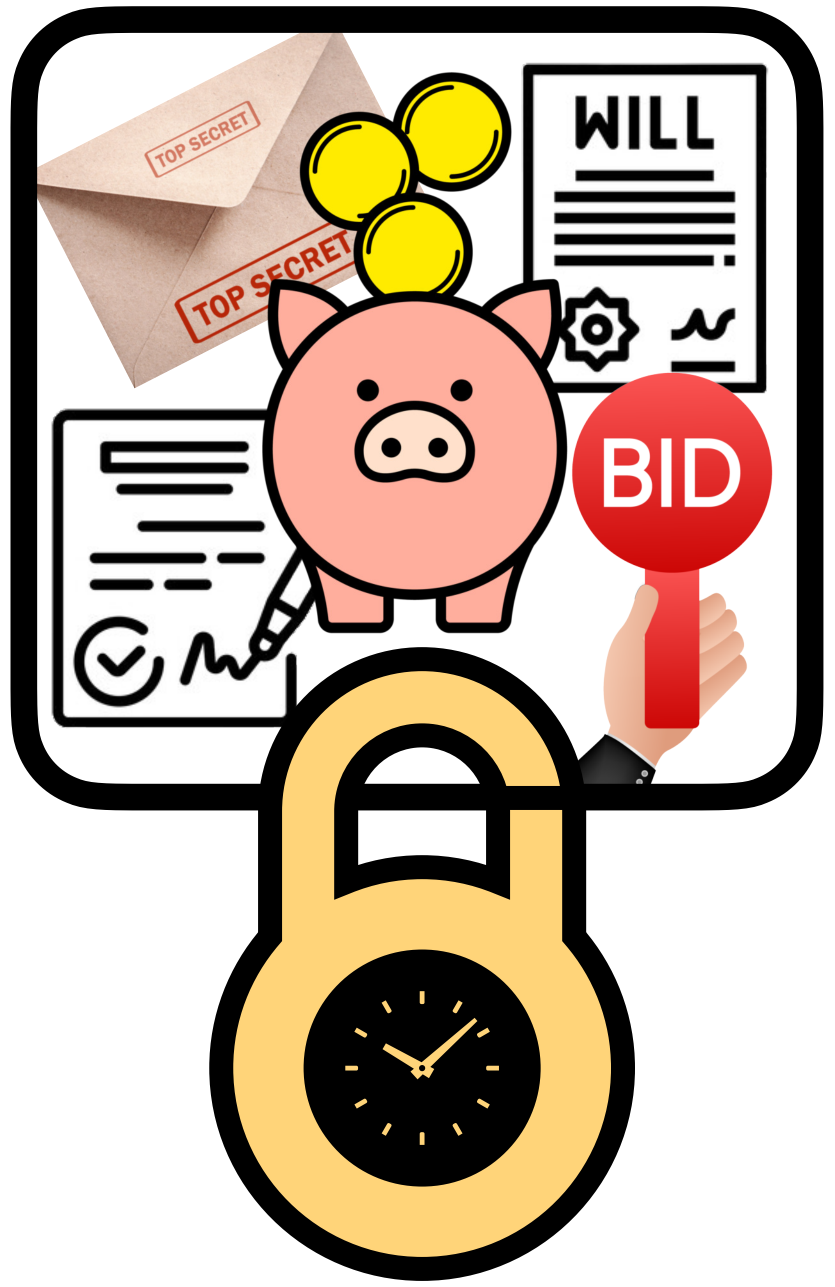
These ideas need more work, but since I have recently mentioned them to Hal Finney, Max More, Mark Miller, and perhaps others, I guess it's time to say something here.

Why would anyone want to send encrypted (sealed) messages into the future?

~~why not?~~

to see if we can?

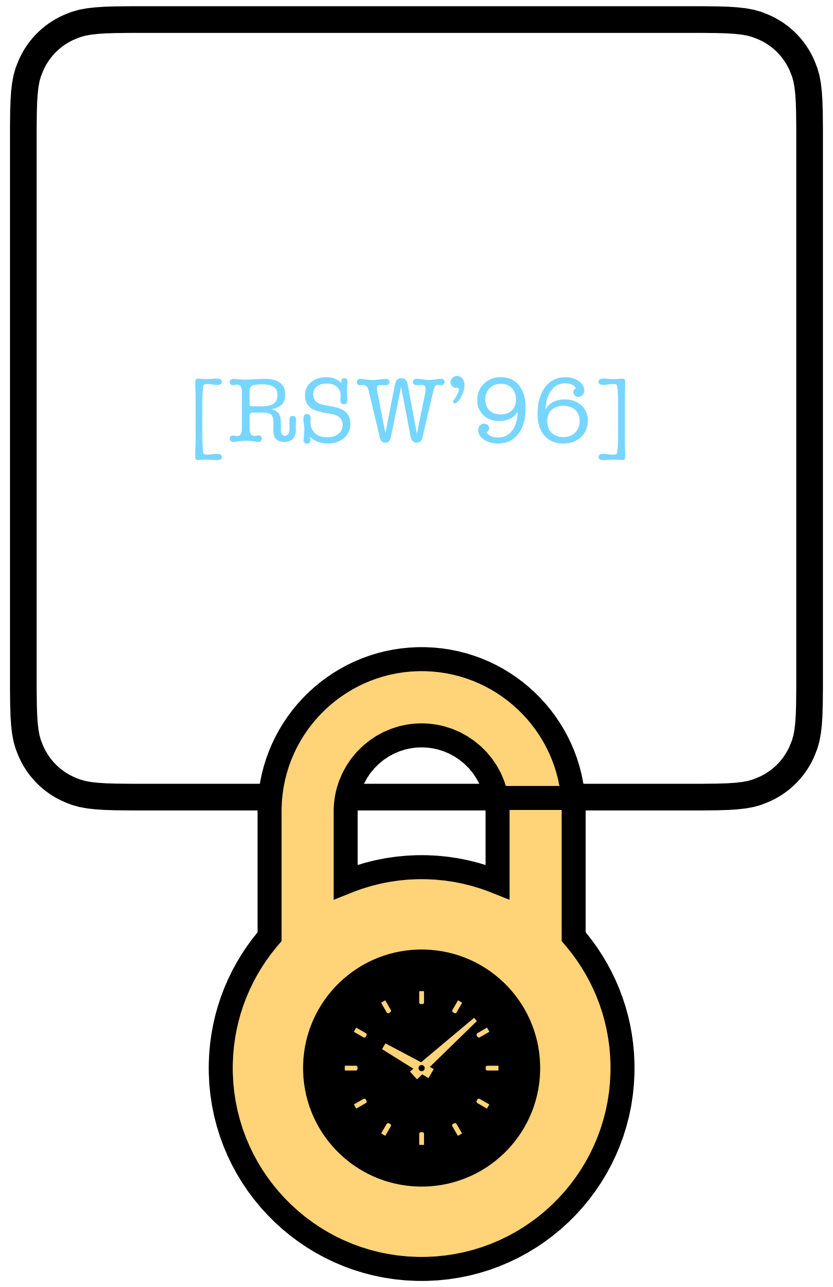
WHY?



- To send money into the future
- To fulfill contracts with long payoff dates
- "In the event of my death…"
- Time-release escrow of, e.g., code.

- Sealed-bid auctions
- Fixed-time embargo for, e.g., legal documents
- Miner extractable value (MEV) prevention

HOW?



Agent-based

$\mathtt{KeyGen}(1^\lambda, t) \longrightarrow (\mathsf{pk}_t, \mathsf{sk}_t)$

The agent generates keys for each time block, publishes all of the public keys $(t, \mathsf{pk}_t)_{t=1,\ldots,T}$ and, for example, either

$\mathtt{Decrypt}(\mathsf{sk}_t, \mathsf{ct}_t) \longrightarrow m$

decrypts ciphertexts at indicated times, or

(t, sk_t)

publishes secret keys at the indicated times.

Puzzle-based

POW [DN'93]
VDF [BBBF'19]

$\mathtt{Gen}(1^\lambda, t) \longrightarrow (\mathsf{puzz}, \mathtt{V})$

A puzzle is generated along with a validation process.
The message is encrypted such that it can be decrypted by a solution.

$$\mathtt{Decrypt}(\mathsf{puzz}, \mathsf{sl}, \mathsf{ct}) \longrightarrow m$$

$\mathtt{Sol}(\mathsf{puzz}) \longrightarrow \mathsf{sl}$

There is an honest solving process, which

$\mathtt{V}(\mathsf{puzz}, \mathsf{sl}) = 1$

(1) outputs solutions that are accepted by the validator, and

(2) is "harder" than puzzle generation, measured by, e.g., RO queries. [MMV'11]

WHAT EVEN IS TIME?

and other comparisons

Agent-based

− synchronicity
the agent knows the time

− trust
the agent is honest

✓ decryption time
not inherently expensive
immediate decryption

✓ modeling
similar to existing primitives

Puzzle-based

− expensive
decryption takes “work”

− modeling
TLP:
- ★ [MMV’11] ROM for sequential computation
- ★ [BDDNO’21] UC + ROM
- ★ [EJTY’23] standard model & leakage

− computation vs. time
assumes a predictable relationship between computation and time

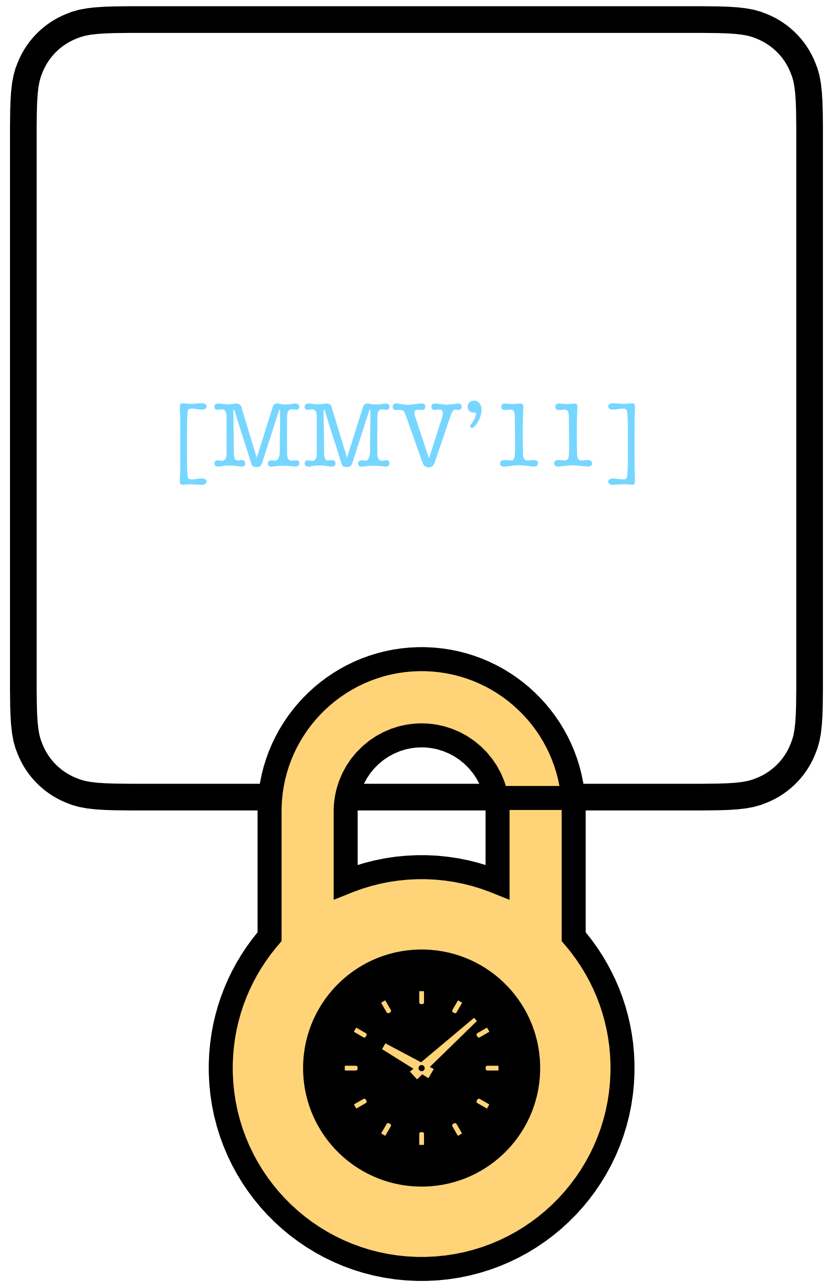
✓ asynchronicity
time starts at decryption

✓ no trust assumption
undesirable TTP not required

− 35y TLP from ’99 solved 15y early
- ★ 3.5y: modern CPU, single core
- ★ 2mo: FPGA hardware

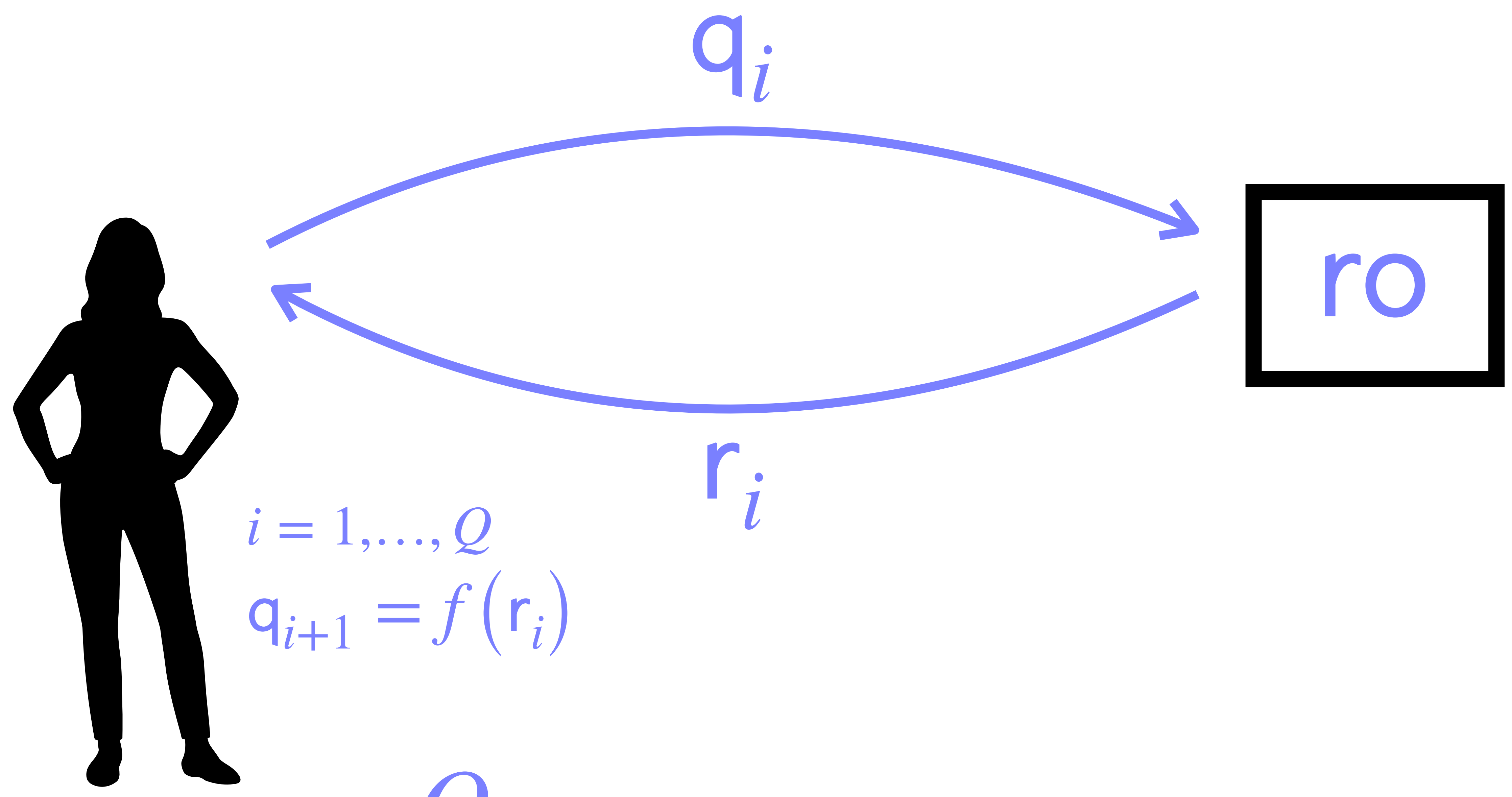
ANOTHER WAY

that I believe deserves attention



Agent-based

Puzzle-based



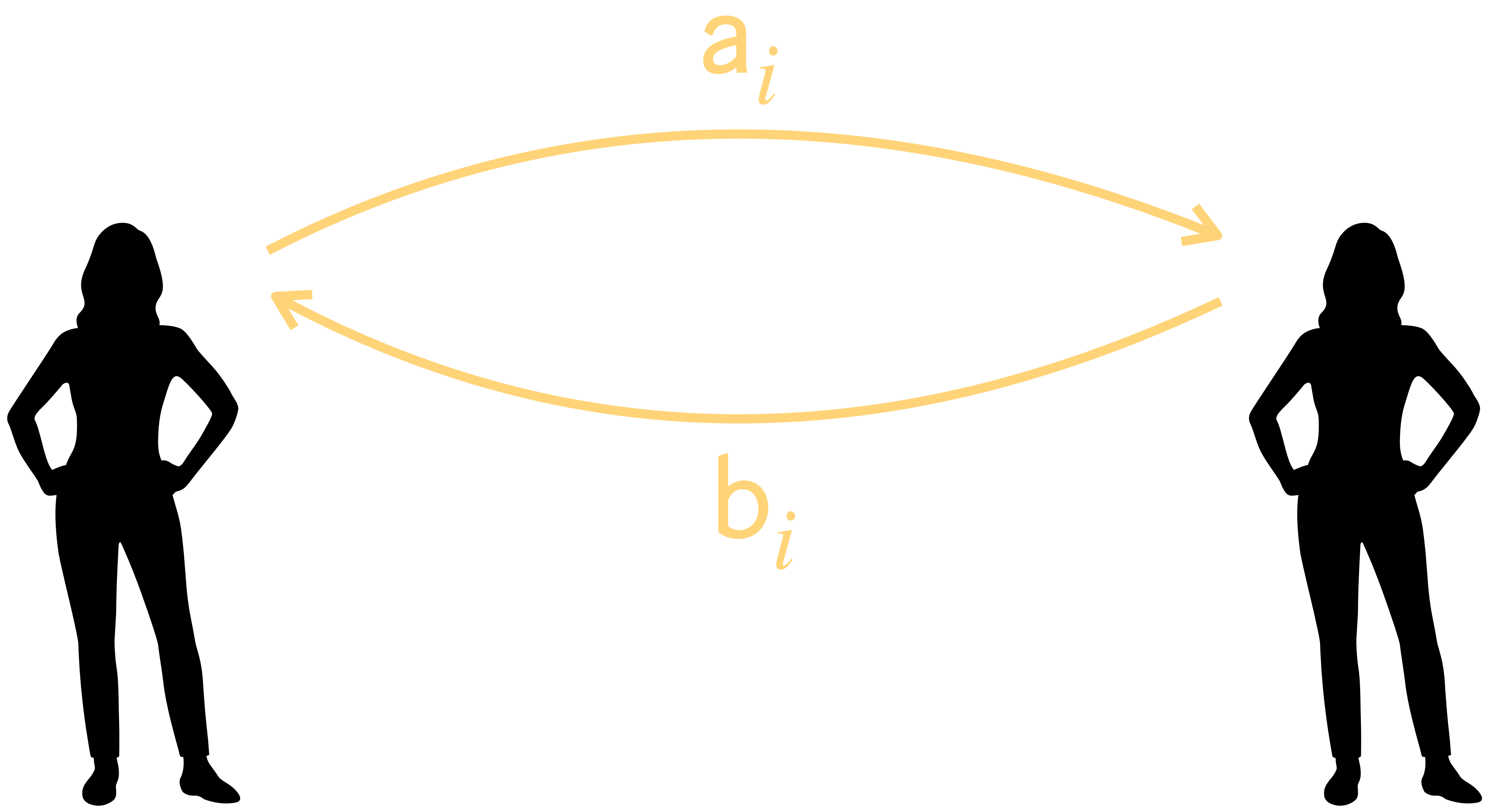
$$\mathsf{Sol}(\mathsf{puzz}) \xrightarrow{Q} \mathsf{sl}$$

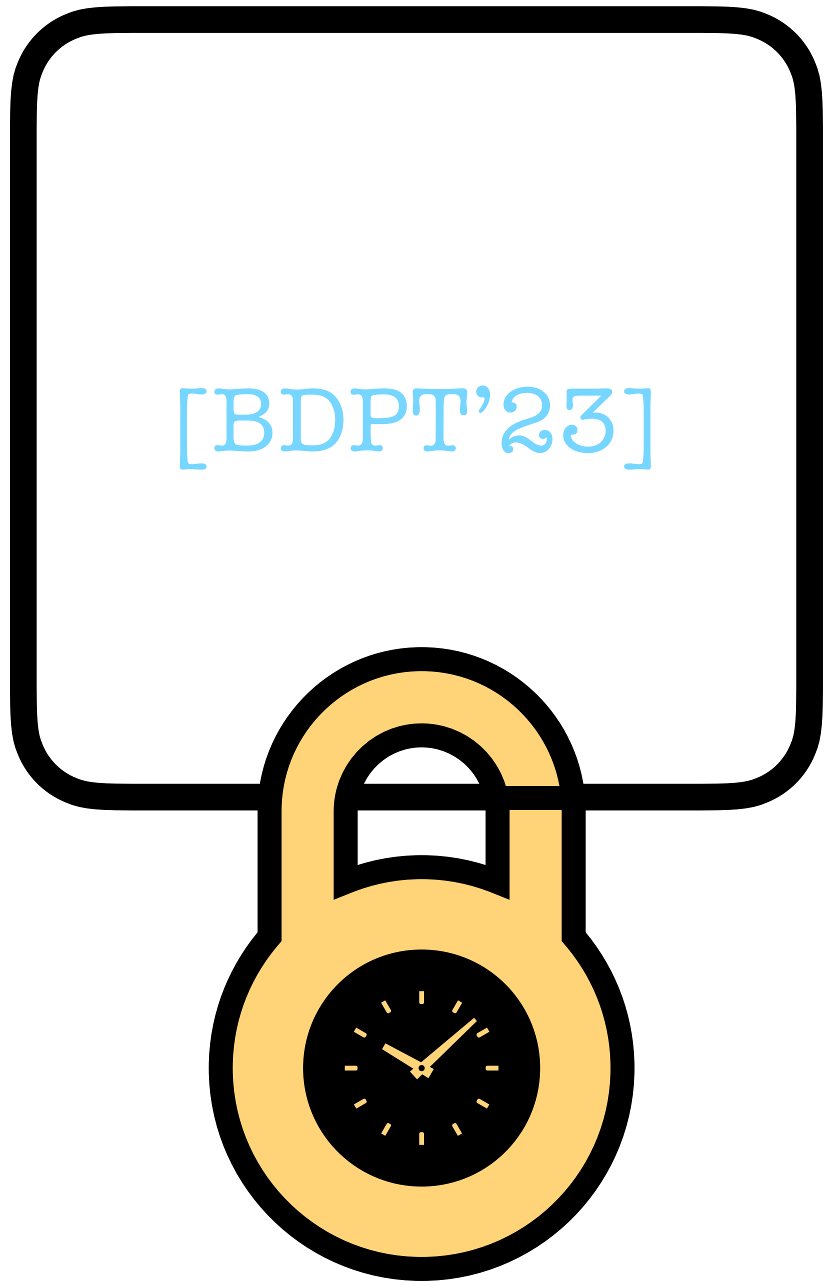
ANOTHER WAY
that I believe deserves attention

Agent-based

Physics-based

Puzzle-based





$a_{i+1} = \text{next-A}\left(i, b_i, sk_A\right)$

$b_i = \text{next-B}\left(i, a_i, sk_B\right)$

$\text{out} := a_Q$

Q vs. time

- ★ Relate to communication, not computation
  - ★ Atomic computation of next-A, next-B
  - ★ Put distance between A, B
  - ★ Lower bound: speed of light

OUR APPROACH

Agent-based

– synchronicity
the agent knows the time

– trust
the agent is honest

✓ decryption time
not inherently expensive
immediate decryption

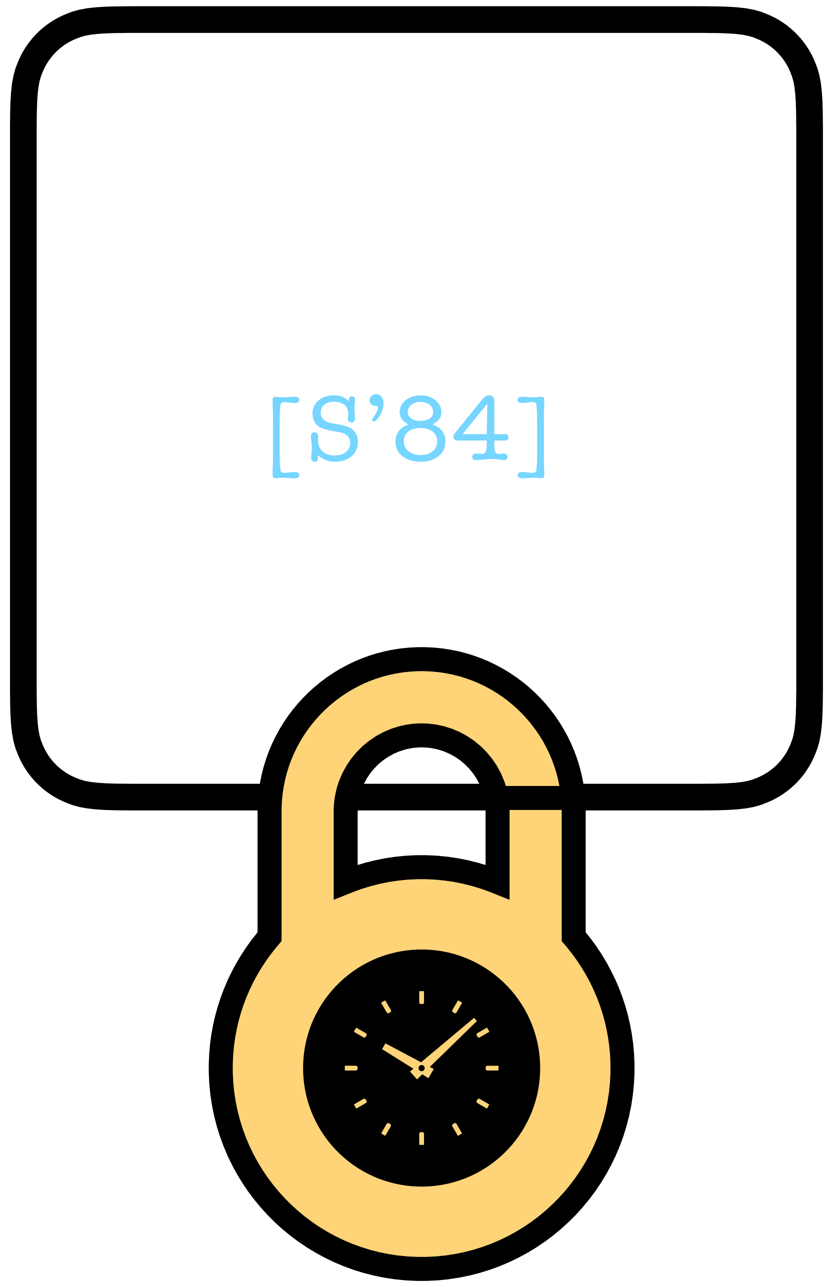
✓ modeling
similar to existing primitives

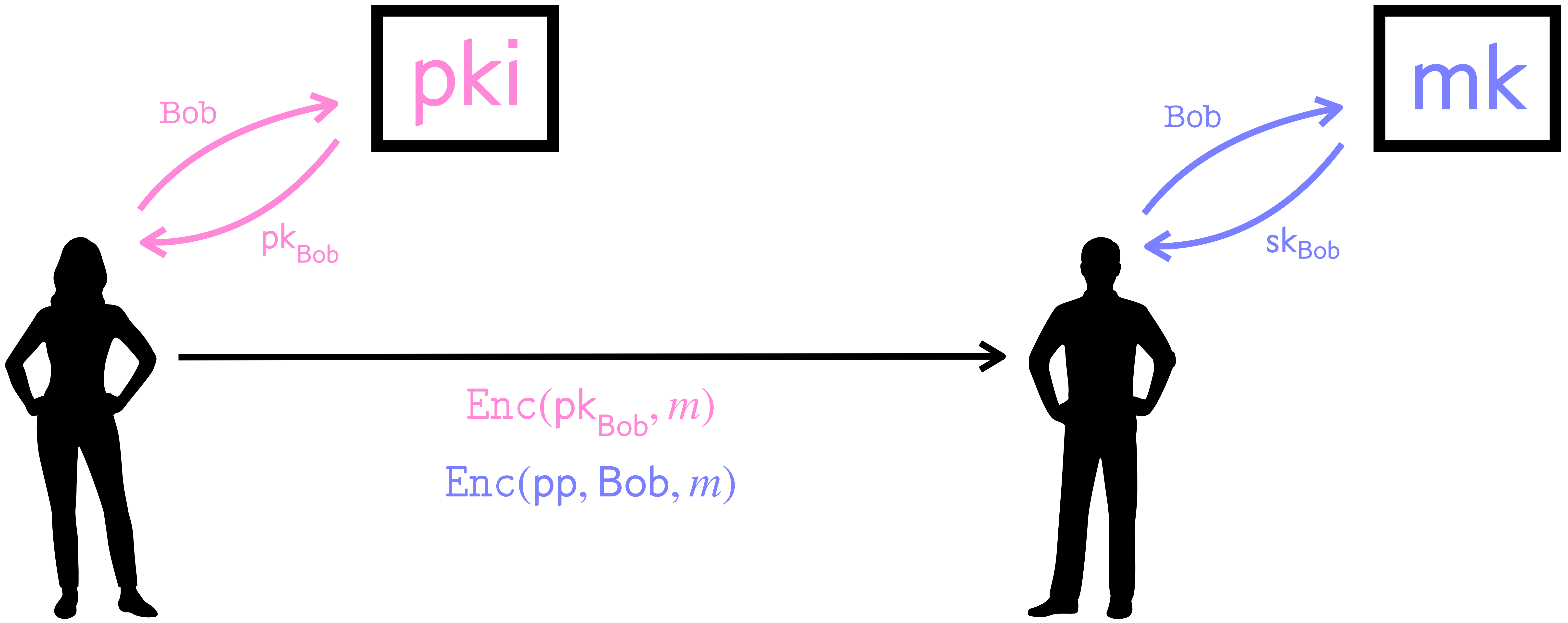
★ IBE implies agent-based TLE

★ distribute the TTP with MPC

ENCRYPTION

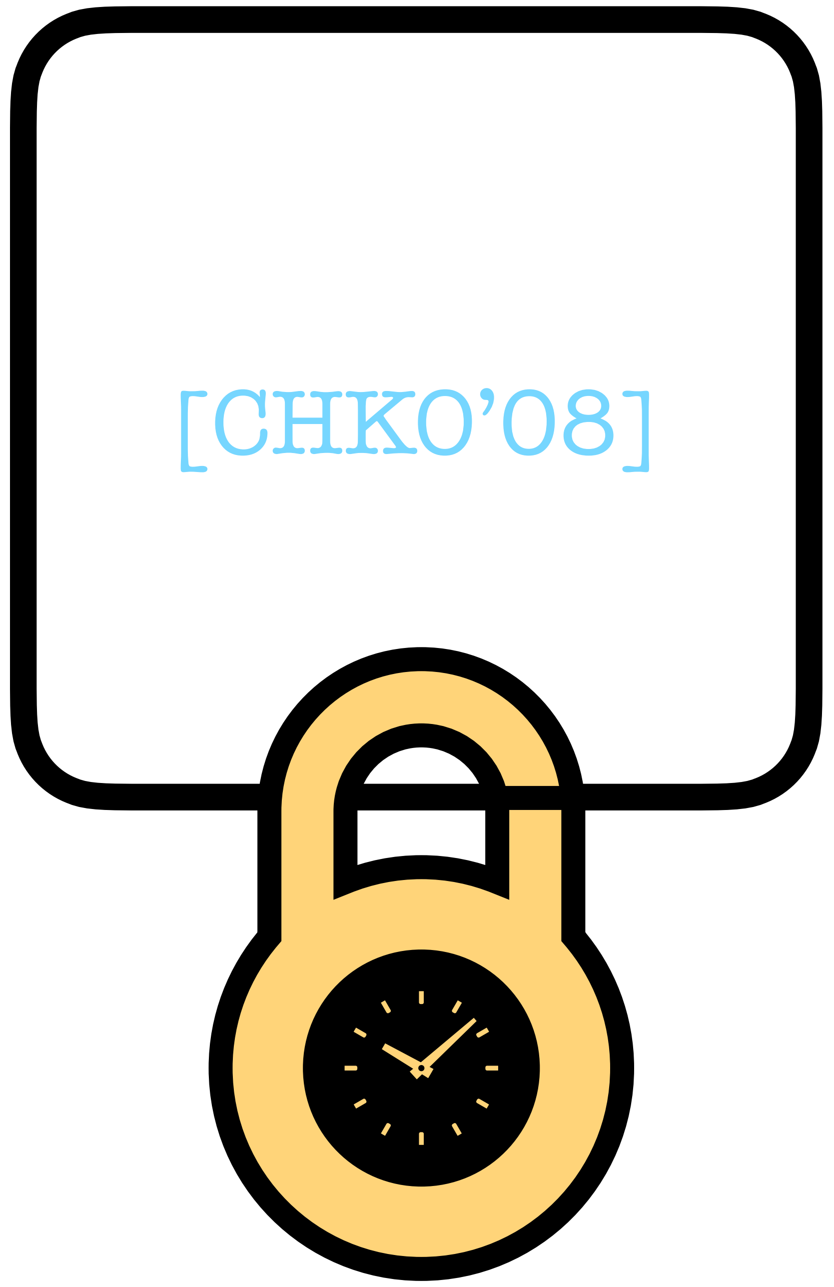
	PKE	IBE
		$1^\lambda \longrightarrow (\mathsf{pp}, \mathsf{mk})$
KeyGen	$1^\lambda \longrightarrow (\mathsf{pk}, \mathsf{sk})$	$(\mathsf{mk}, \mathsf{id}) \longrightarrow \mathsf{sk}_{\mathsf{id}}$
Encrypt	$(\mathsf{pk}, m) \longrightarrow \mathsf{ct}$	$(\mathsf{pp}, \mathsf{id}, m) \longrightarrow \mathsf{ct}_{\mathsf{id}}$
Decrypt	$(\mathsf{sk}, \mathsf{ct}) \longrightarrow m$	$(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) \longrightarrow m$



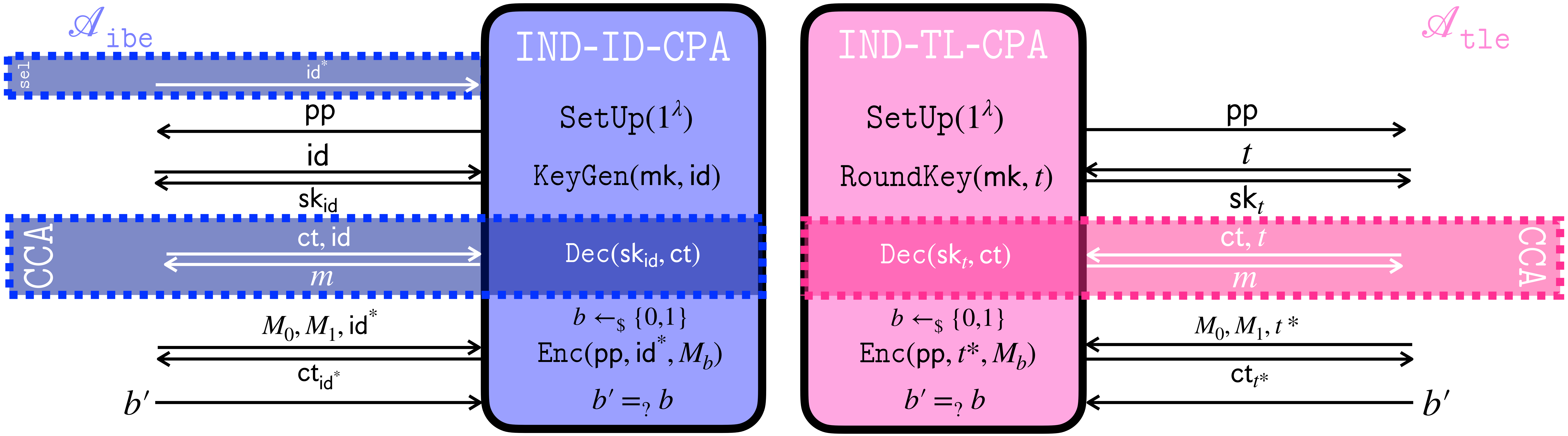


OUR APPROACH

★ IBE implies agent-based TLE



IBE		TLE	
SetUp	$1^\lambda \longrightarrow (\mathsf{pp}, \mathsf{mk})$	SetUp	$1^\lambda \longrightarrow (\mathsf{pp}, \mathsf{mk})$
KeyGen	$(\mathsf{mk}, \mathsf{id}) \longrightarrow \mathsf{sk}_{\mathsf{id}}$	RoundKey	$(\mathsf{mk}, t) \longrightarrow \mathsf{sk}_t$
Encrypt	$(\mathsf{pp}, \mathsf{id}, m) \longrightarrow \mathsf{ct}_{\mathsf{id}}$	Encrypt	$(\mathsf{pp}, t, m) \longrightarrow \mathsf{ct}_t$
Decrypt	$(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}_{\mathsf{id}}) \longrightarrow m$	Decrypt	$(\mathsf{sk}_t, \mathsf{ct}_t) \longrightarrow m$



[CV’19]

Timelock Encryption: an Overview and Retrospective

Presented at NIST-STPPA7 on 2025-01-16

Kelsey Melissaris, **Yolan Romailler**

RANDAMU

What we want

Encrypt something to the future

What we have

How to?

A paragraph in the original BF-IBE paper in 2001 mentions that identity decryption keys can be used as *signatures*, BLS does that.
BLS signatures can be seen as **decryption keys** for a **specific identity**.

Intuition

BLS reminder

Basically we are using the fact that the pairing operation is bilinear to *extract the secret key* once from the public key and once from the signature to perform a key agreement:

$$\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

Signature

$$\begin{aligned} \mathrm{e}(G_1, \boxed{\pi}) &= \mathrm{e}(G_1, sM) = \boxed{s\,\mathrm{e}(G_1, M)} \\ \mathrm{e}(\boxed{P_g}, M) &= \mathrm{e}(sG_1, M) = \boxed{s\,\mathrm{e}(G_1, M)} \end{aligned}$$

Public key

Two different ways to obtain the same target group element

Intuition

Use it for encryption

To get secrecy, we need to add the notion of ephemeral key to the mix:

$$P_e = rG_1, r \in \{0,1\}^\ell$$

$$r\,\mathrm{e}(P_g, M) = r\,\mathrm{e}(sG_1, M) = \boxed{rs\,\mathrm{e}(G_1, M)}$$

$$\mathrm{e}(P_e, \pi) = \mathrm{e}(rG_1, sM) = \boxed{rs\,\mathrm{e}(G_1, M)}$$

What we have

How to?

We happen to have a live production network issuing random beacons signed using **threshold BLS signatures** at a fixed frequency: drand.

We can design a threshold agent-based timelock scheme with it.

Improving on prior art

Decentralization

- While the idea of using pairing-based systems for timelock isn’t new, the way we have transformed it into a practical system people are ready to trust is by decentralizing the trust requirement.
- **BLS** signatures are very easy to “thresholdize”, and so is threshold IBE!
- By avoiding to rely on single trusted parties, we can easily build and deploy a threshold Timelock Encryption (tTLE) system in practice that people can trust.

What we have

drand

- drand is an **open source** software in Go ran by a set of independent nodes that collectively produce beacons.
- Provides **public, verifiable random beacons** using
  - **Threshold BLS** on the curve **BLS12-381**
  - Pedersen Distributed Key Generation and resharings
- Tested, **audited**, and deployed at scale by the League of Entropy since 2019. Used in production since 2020.

What we have

The League of Entropy

18+ organizations

4M+ rounds

18 nodes

4+ years

5 endpoints

0 disruptions since launch in 2020-08-10

0 single point of failure

LEAGUE OF ENTROPY

CLOUDFLARE

DIA

The Initiative for CryptoCurrencies and Contracts

UPC

Universidad de Chile

KUDELSKI SECURITY

ChainSafe

UCL

Protocol Labs

EPFL

AUTOMATA

C4DT

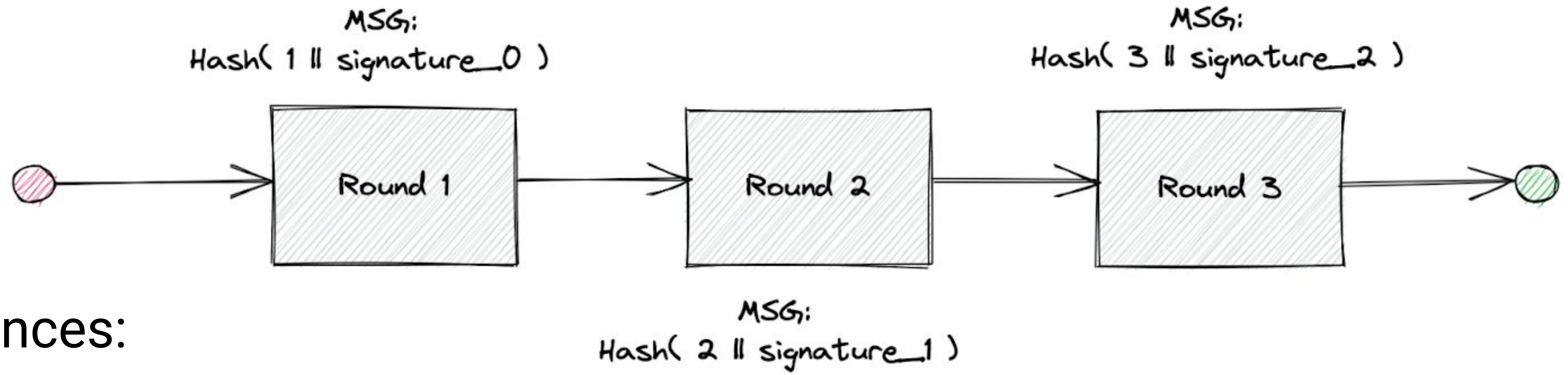
What we have

drand beacons map to a precise time!

In practice

Problem: chained randomness

The beacons on the LoE mainnet were Chained



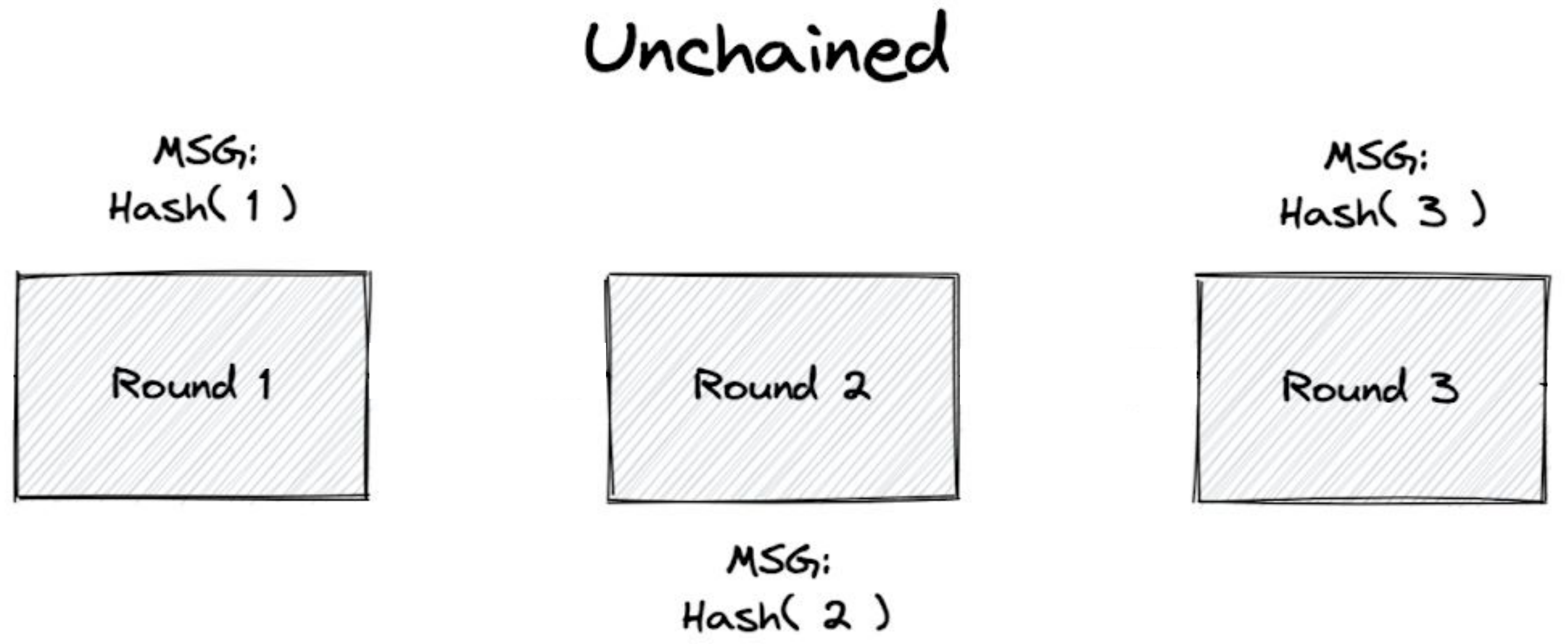
Consequences:

- **No one** knows the round message more than **one round** in advance
  - e.g. Hash(3 || signature_2) can only be known at round 2
- Requires the full chain for proper full verification
- Not compatible with IBE-based Timelock

In practice

Solution: Unchained Randomness

New **unchained randomness** mode introduced in February 2022, launched on Testnet in May and used in production since August 2023.



Consequences:

- Messages are mapped to a given time: Hash(10) happens at time T_10
- Everybody **knows the future round message** getting signed ahead of time.
- Verification is much simpler and stateless, without impacting trust/security.

In practice

Problem: performance/size tradeoff

BLS signatures on BLS12-381 done on $\mathbb{G}_2$ are ~96 bytes in compressed form.

Furthermore we need to map the message M to the group $\mathbb{G}_2$, which is at least 10x more costly than doing so on $\mathbb{G}_1$.

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

BLS signature

$$\mathrm{e}(G_1, \pi) = \mathrm{e}(G_1, \boxed{sM}) = s\,\mathrm{e}(G_1, M)$$
$$\mathrm{e}(P_g, M) = \mathrm{e}(\boxed{sG_1}, M) = s\,\mathrm{e}(G_1, M)$$

BLS public key

In practice

Solution: swap G1 and G2

New **swapped group scheme** launched in February 2023.

BLS signature

$$e(\pi, G_2) = e(\boxed{sM}, G_2) = s\ e(M, G_2)$$

$$e(M, P) = e(M, \boxed{sG_2}) = s\ e(M, G_2)$$

BLS public key

Storage benefit: signatures are now 50% smaller at 48 bytes vs 96 bytes!

In practice

Digression: hybrid encryption

We can only encrypt small blocks of data using our tTLE scheme, since we opted for using a hash for key derivation rather than a XOF... so we rely in practice on **hybrid encryption** to encrypt larger chunks of data.

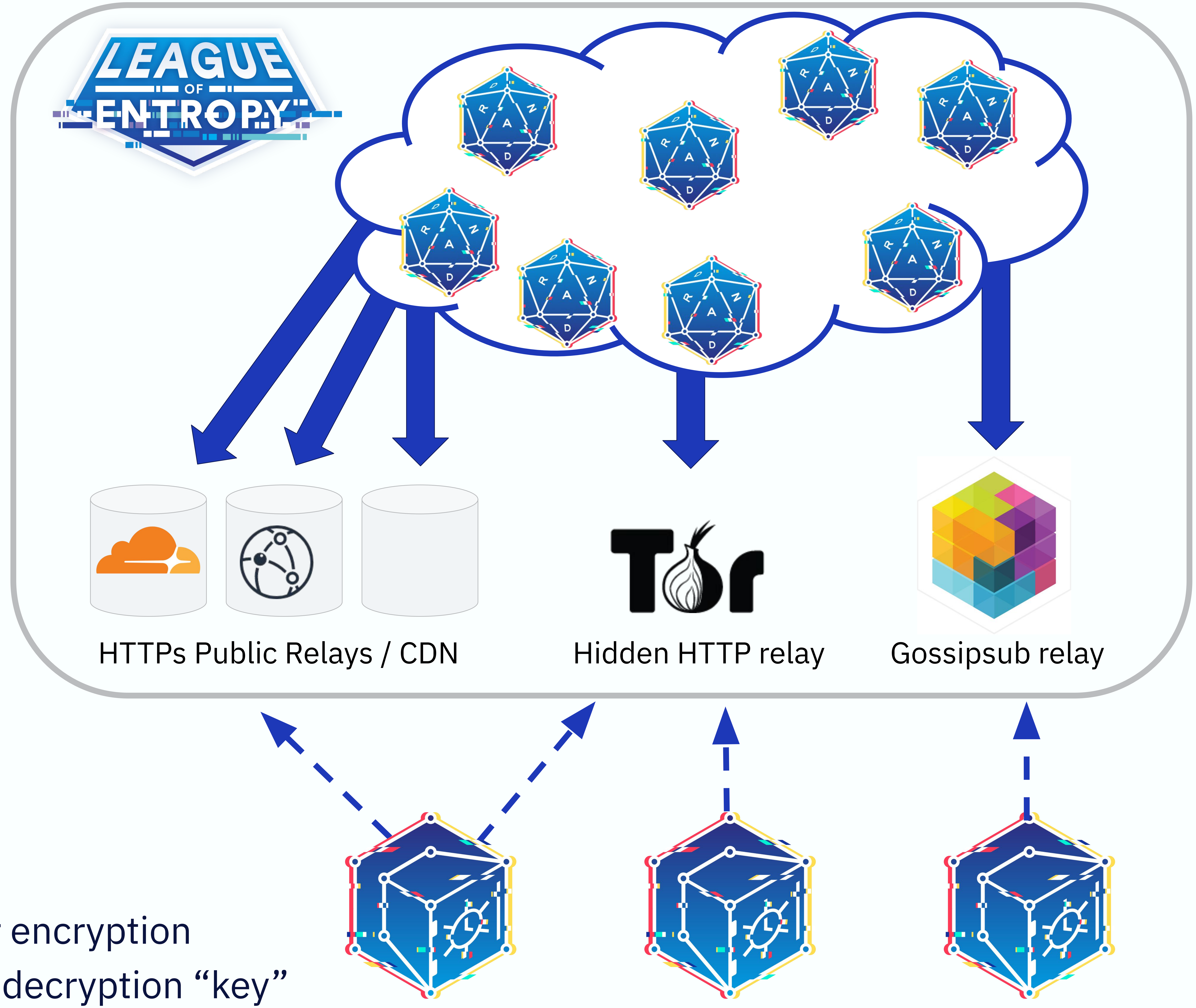
For ease, we used age to achieve this using a custom stanza for timelock and delegating key-wrapping and data encryption to it. In theory in a way compatible with its new plugin system:

```
age-encryption.org/v1
-> tlock 764081 dbd506d6ef76e5f386f41c651dcb808c5bcbd75471cc4eafa3f4df7ad4e4c493
```

In practice
Our timelock

The League of Entropy part

- Permissioned network
- Threshold t > (n / 2) + 1
- **100% uptime** since mainnet launch in 2020
- **Stable group public key**
- Granularity of 3s
- Solid Distribution Network
- Is not dedicated for timelock



The Timelock part

- Client-side only operations
- Needs the group public key for encryption
- Queries the drand network for decryption “key”

Details

It’s almost all on ePrint

tlock: practical timelock encryption from threshold BLS

Nicolas Gailly[1], Kelsey Melissaris[2], Yolan Romailler[1]

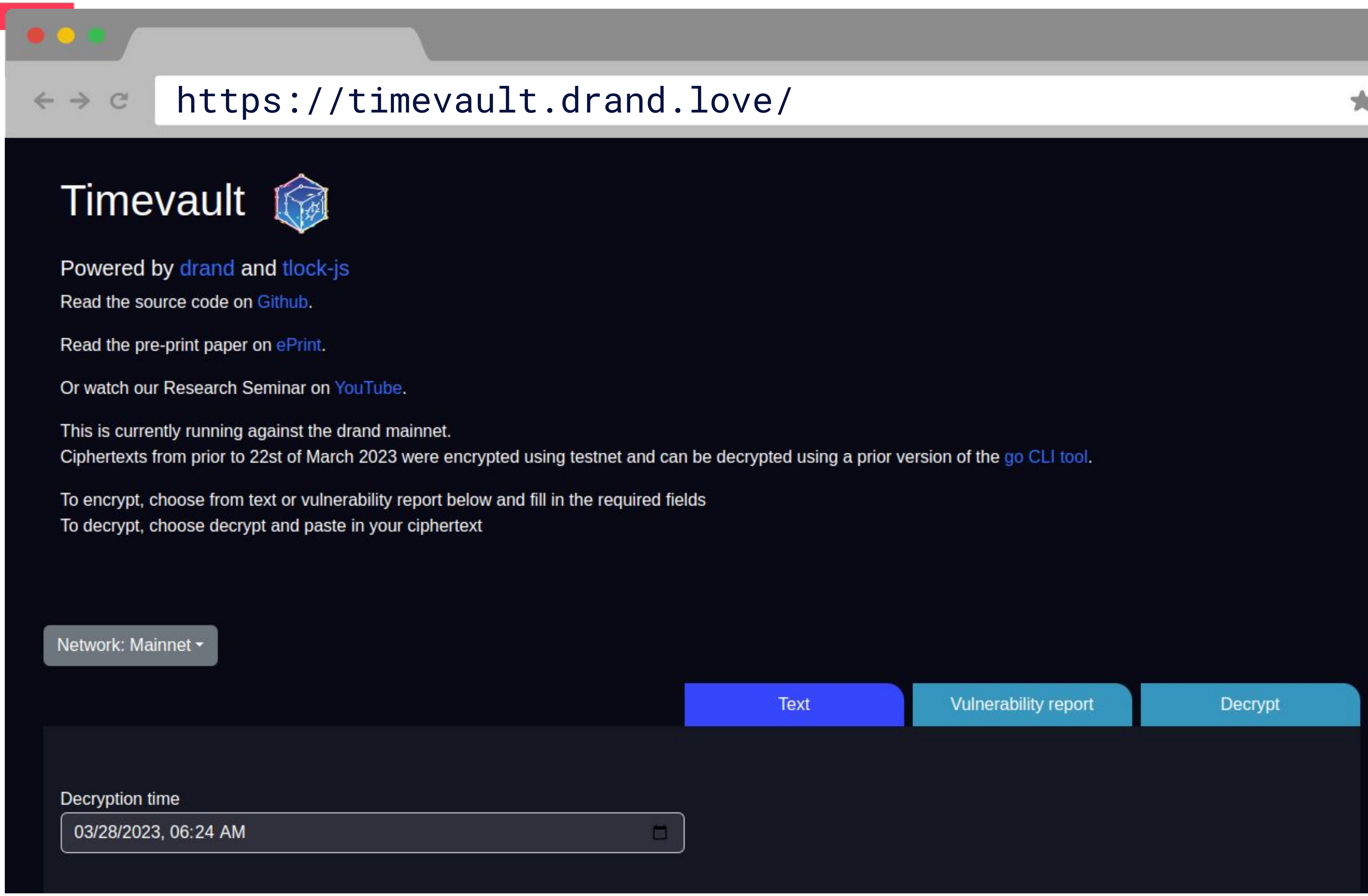
[1] Protocol Labs
https://research.protocol.ai/
[2] Department of Computer Science
Aarhus University, Denmark

Abstract. We present a practical construction and implementation of timelock encryption, in which a ciphertext is guaranteed to be decryptable only after some specified time has passed. We employ an existing threshold network, the League of Entropy, implementing threshold BLS [BLS01, Bol03] in the context of Boneh and Franklin’s identity-based encryption [BF01] (BF-IBE). At present this threshold network broadcasts BLS signatures over each round number, equivalent to the current time interval, and as such can be considered a decentralised key holder periodically publishing private keys for the BF-IBE where identities are the round numbers. A noticeable advantage of this scheme is that only the encryptors and decryptors are required to perform any additional cryptographic operations; the threshold network can remain unaware of these computations and does not have to change to support the scheme. We also release an open-source implementation of our scheme and a live web page that can be used in production now relying on the existing League of Entropy network acting as a distributed public randomness beacon service using threshold BLS signatures.

This work is explained in more detail in our ePrint paper, and we are looking into UC security proofs and extending it a bit more, so don’t hesitate to check it out:
https://ia.cr/2023/189

So, let’s look at the “Real World” part of it that’s not on ePrint! What does “practical” mean and why are we here today?

Try it live:



timevault.drand.love

What’s next?

Future work

- Looking into doing ZKPs on the timelocked input for specific usecases.
- Implement more use cases! (Sealed bid auctions, MEV prevention, etc.)
- Look into “PQ-IBE” schemes, there are LWE based IBE ones.
- Look into “threshold post-quantum signatures”!
- Most ecosystems don’t have BLS12-381 built-in functions (need for a spec?)
- Some implementations do not yet support signatures on $\mathbb{G}_1$.
- The League of Entropy is welcoming new members!

Thank you !

For more information and/or if you want to reach out, go to:

https://github.com/drand/tlock

https://github.com/drand/tlock-js

https://drand.love/blog/

yolan@randa.mu
@anomalroil

RANDAMU