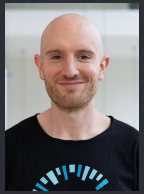
Transparent Batchable Time-lock Puzzles and Applications to Byzantine Consensus

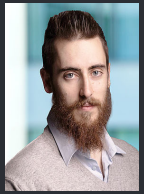
Shravan Srinivasan

UMD



Julian Loss

CISPA

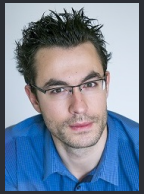


Giulio Malavolta

MPI-SP

Kartik Nayak

Duke



Charalampos Papamanthou

Yale

Sri AravindaKrishnan Thyagarajan

NTT Research

PKC ’23

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”

m

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”
- Fast puzzle generation

Time-Lock Puzzles (TLP) [May92, RSW96]

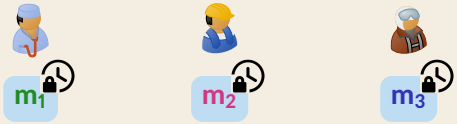
- Encrypt a message “to the future”
- Fast puzzle generation
- Security against parallel adversaries

Time-Lock Puzzles (TLP) [May92, RSW96]

- Encrypt a message “to the future”
- Fast puzzle generation
- Security against parallel adversaries
- Applications in auctions, blockchains, timed-commitments, and more

Batchable Time-Lock Puzzles [MT19, TBM+20]

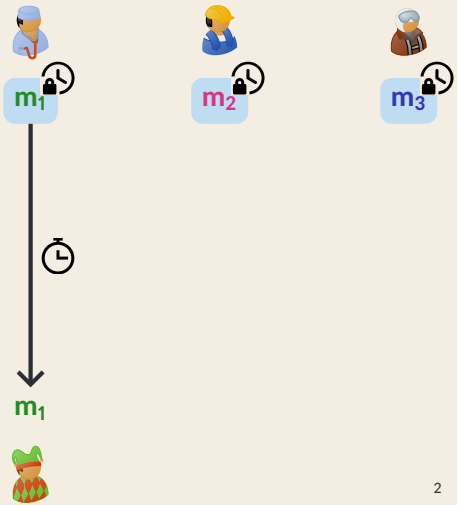
Traditional TLPs [RSW96]:



Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

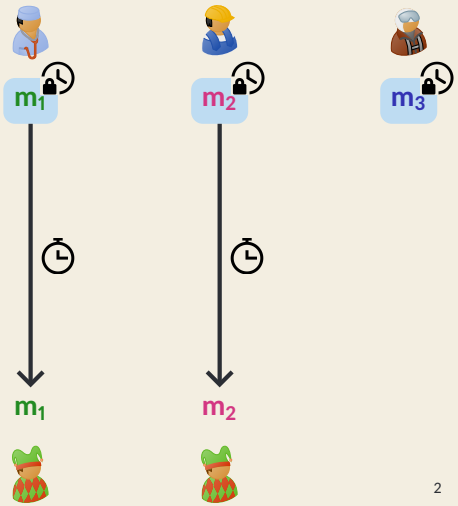
- Doesn't scale with users



Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

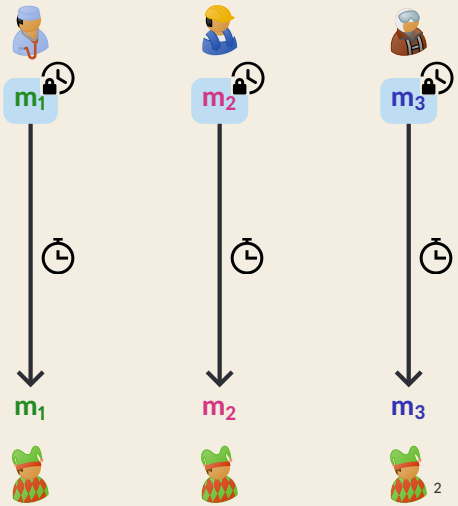
- Doesn't scale with users
- Need to solve puzzles individually



Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

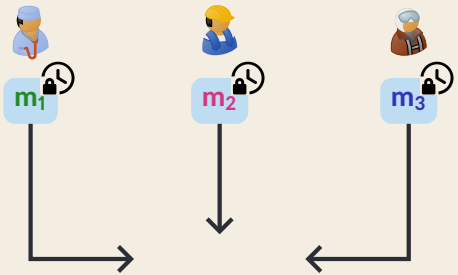
- Doesn't scale with users
- Need to solve puzzles individually



Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

- Doesn't scale with users
- Need to solve puzzles individually



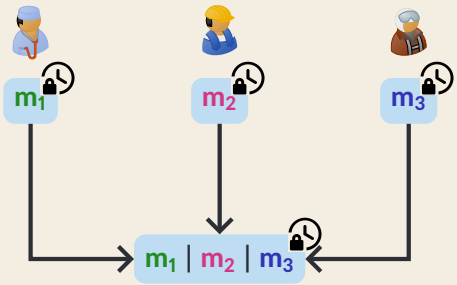
Batchable TLPs [MT19, TBM+20]:

- Open multiple puzzles for the price of one
- Supports homomorphism

Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

- Doesn't scale with users
- Need to solve puzzles individually



Batchable TLPs [MT19, TBM+20]:

- Open multiple puzzles for the price of one
- Supports homomorphism

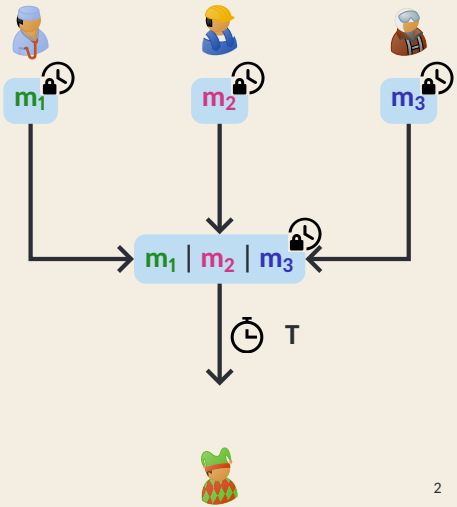
Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

- Doesn't scale with users
- Need to solve puzzles individually

Batchable TLPs [MT19, TBM+20]:

- Open multiple puzzles for the price of one
- Supports homomorphism



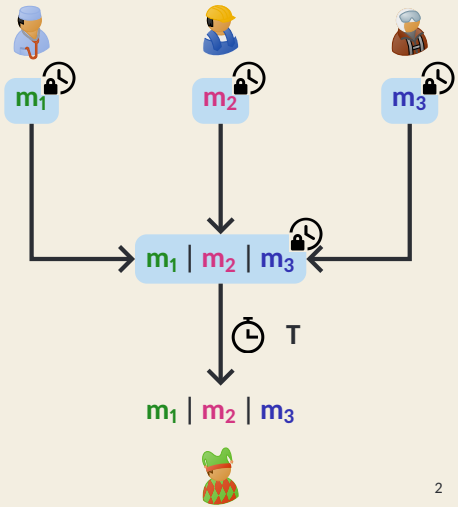
Batchable Time-Lock Puzzles [MT19, TBM+20]

Traditional TLPs [RSW96]:

- Doesn't scale with users
- Need to solve puzzles individually

Batchable TLPs [MT19, TBM+20]:

- Open multiple puzzles for the price of one
- Supports homomorphism



Problem Statement

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
RSA-based [TBM+20, MT19]	✓	×	×	×
Class-groups based [TCLM21]	✓	✓	×	×

Problem Statement

Unbounded batching

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
RSA-based [TBM+20, MT19]	✓	×	×	×
Class-groups based [TCLM21]	✓	✓	×	×

Problem Statement

Unbounded batching

Individual puzzle size independent of the batch size

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
RSA-based [TBM+20, MT19]	✓	×	×	×
Class-groups based [TCLM21]	✓	✓	×	×

Problem Statement

Unbounded batching

Individual puzzle size independent of the batch size

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
RSA-based [TBM+20, MT19]	✓	×	×	×
Class-groups based [TCLM21]	✓	✓	×	×
?	✓	✓	✓	✓

Why do we care?

Useful in the decentralized setting [TBM+20, WXDS20, tez22]:

- Impractical to rely on trusted setup
- Unknown number of nodes
- Large puzzles increases communication overhead

Contributions

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
This work	✓	✓	✓	✓

Contributions

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
This work	✓	✓	✓	✓

Uses indistinguishability obfuscation

Contributions

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
This work	✓	✓	✓	✓

Uses indistinguishability obfuscation

Applications in consensus:

Contributions

Scheme	Batchable	Transparent setup	No apriori bound	Compact puzzles
This work	✓	✓	✓	✓

Uses indistinguishability obfuscation

Applications in consensus:

- First permissionless protocol in the *mobile sluggish* model
- First expected $O(1)$-round Byzantine broadcast under strongly adaptive and corrupt majority setting

Outline

1. Preliminaries

2. TLP construction
   - 2.1 Puzzle generation
   - 2.2 Batch Solving

3. Application: Permissionless Consensus
   - 3.1 Network model
   - 3.2 Protocol

Outline

1. Preliminaries

2. TLP construction
    - 2.1 Puzzle generation
    - 2.2 Batch Solving

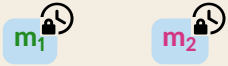
3. Application: Permissionless Consensus
    - 3.1 Network model
    - 3.2 Protocol

Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:

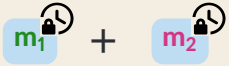
Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



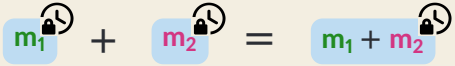
Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



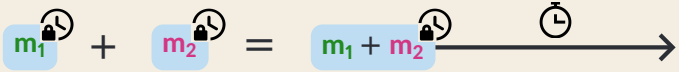
Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



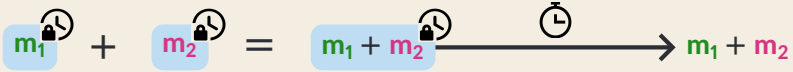
Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



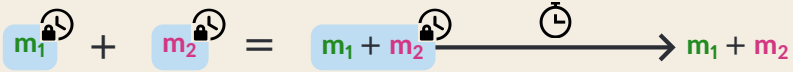
Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



Preliminaries

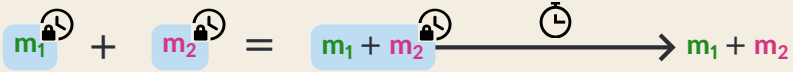
Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



Key-Homomorphic Pseudorandom Functions (KH-PRF) [BV15]:

Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:

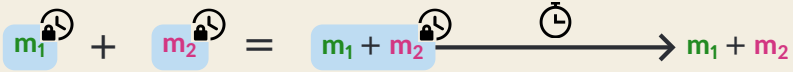


Key-Homomorphic Pseudorandom Functions (KH-PRF) [BV15]:

Informally, a (puncturable) pseudorandom function, F, is KH-PRF, if

Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



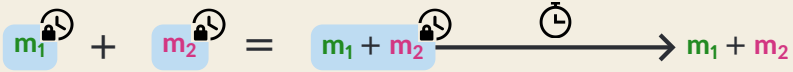
Key-Homomorphic Pseudorandom Functions (KH-PRF) [BV15]:

Informally, a (puncturable) pseudorandom function, F, is KH-PRF, if

$$F(k_1, m) + F(k_2, m) \approx F(k_1 + k_2, m)$$

Preliminaries

Homomorphic Time-Lock Puzzles (HTLP) [MT19, TCLM21]:



Key-Homomorphic Pseudorandom Functions (KH-PRF) [BV15]:

Informally, a (puncturable) pseudorandom function, F, is KH-PRF, if

$$F(k_1, m) + F(k_2, m) \approx F(k_1 + k_2, m)$$

for all keys k_1, k_2 and all message m.

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input
  - Has the message $\mathbf{m_i}$ and the PRF key $\mathbf{k_i}$ hardwired

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input
  - Has the message $\mathbf{m_i}$ and the PRF key $\mathbf{k_i}$ hardwired
  - Returns HTLP of $\mathbf{k_i}$

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input
  - Has the message $\mathbf{m_i}$ and the PRF key $\mathbf{k_i}$ hardwired
  - Returns HTLP of $\mathbf{k_i}$

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input
  - Has the message $\mathbf{m_i}$ and the PRF key $\mathbf{k_i}$ hardwired
  - Returns HTLP of $\mathbf{k_i}$ and *structured vector* of ciphertexts

Puzzle Generation

- Player **i** sends an obfuscated program $\mathbf{P_i}$
- Program $\mathbf{P_i}$:
  - Takes the batch size **n** as input
  - Has the message $\mathbf{m_i}$ and the PRF key $\mathbf{k_i}$ hardwired
  - Returns HTLP of $\mathbf{k_i}$ and *structured vector* of ciphertexts

$$\begin{bmatrix} F(k_i, 1) \\ \vdots \\ F(k_i, i) \\ \vdots \\ F(k_i, n) \end{bmatrix} + m_i$$

Batch Solving

Obf. prog. P_1 P_2 P_3

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP 

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix}$$

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP 

Masked cipher txt. $\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix}$

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP k_1 k_2

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} \quad \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix}$$

Batch Solving

Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP k_1 k_2

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} \quad \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP k_1 k_2 k_3

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} \quad \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} \quad \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP k_1 + k_2 + k_3

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} + \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} + \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} + \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} + \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} + \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} + \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix} = \begin{bmatrix} \sum_i F(k_i, 1) + m_1 \\ \\ \\ \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} + \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} + \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix} = \begin{bmatrix} \sum_i F(k_i, 1) + m_1 \\ \sum_i F(k_i, 2) + m_2 \\ \end{bmatrix}$$

Batch Solving



Batch size $n = 3$

Obf. prog. P_1 P_2 P_3

HTLP $k_1 + k_2 + k_3 = k^* = \sum_i k_i$

Masked cipher txt.

$$\begin{bmatrix} F(k_1, 1) + m_1 \\ F(k_1, 2) \\ F(k_1, 3) \end{bmatrix} + \begin{bmatrix} F(k_2, 1) \\ F(k_2, 2) + m_2 \\ F(k_2, 3) \end{bmatrix} + \begin{bmatrix} F(k_3, 1) \\ F(k_3, 2) \\ F(k_3, 3) + m_3 \end{bmatrix} = \begin{bmatrix} \sum_i F(k_i, 1) + m_1 \\ \sum_i F(k_i, 2) + m_2 \\ \sum_i F(k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$k^* = \sum_i k_i$$

$$\begin{bmatrix} \sum_i F(k_i, 1) + m_1 \\ \sum_i F(k_i, 2) + m_2 \\ \sum_i F(k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$k^* = \sum_i k_i$$

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ \sum_i F(k_i, 2) + m_2 \\ \sum_i F(k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$k^* = \sum_i k_i$$

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ \sum_i F(k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$k^* = \sum_i k_i$$

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} \qquad \begin{bmatrix} F(k^*, 1) \\ \\ \\ \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} \quad \begin{bmatrix} F(k^*, 1) \\ F(k^*, 2) \\ \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} \quad \begin{bmatrix} F(k^*, 1) \\ F(k^*, 2) \\ F(k^*, 3) \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} - \begin{bmatrix} F(k^*, 1) \\ F(k^*, 2) \\ F(k^*, 3) \end{bmatrix} = \begin{bmatrix} m_1 \\ \\ \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} - \begin{bmatrix} F(k^*, 1) \\ F(k^*, 2) \\ F(k^*, 3) \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \end{bmatrix}$$

Batch Solving (contd.)

$$\begin{bmatrix} F(\sum_i k_i, 1) + m_1 \\ F(\sum_i k_i, 2) + m_2 \\ F(\sum_i k_i, 3) + m_3 \end{bmatrix} - \begin{bmatrix} F(k^*, 1) \\ F(k^*, 2) \\ F(k^*, 3) \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$$

Outline

Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Nakamoto consensus:

Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

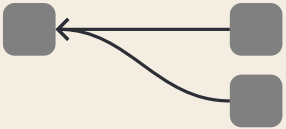
Nakamoto consensus:

Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Nakamoto consensus:

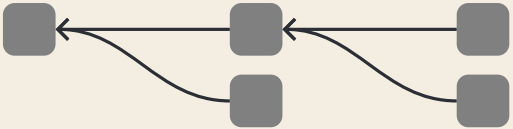


Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Nakamoto consensus:

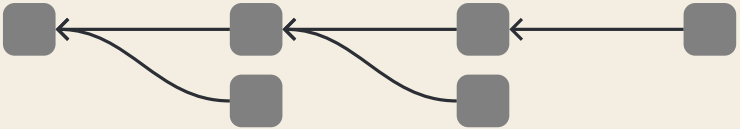


Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Nakamoto consensus:



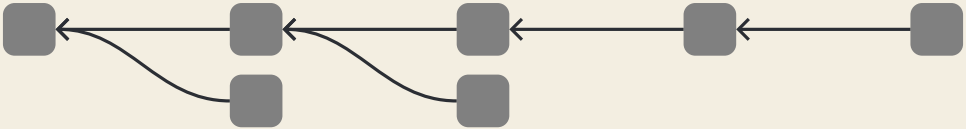
Secure in the synchronous model [GKL15, PSS17, LG19]

Application: Permissionless Consensus

Permissionless setting:

- Exact number of nodes unknown
- No authentication mechanism
- Communication over unauthenticated channels

Nakamoto consensus:



Secure in the synchronous model [GKL15, PSS17, LG19]

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous
Honest msg. delay	Known Δ

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous
Honest msg. delay	Known Δ
Permissionless	

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Partially/Asynchronous
Honest msg. delay	Known Δ	
Permissionless		

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Partially/Asynchronous
Honest msg. delay	Known Δ	No known Δ
Permissionless		

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Partially/Asynchronous
Honest msg. delay	Known Δ	No known Δ
Permissionless		

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Sluggish	Partially/Asynchronous
Honest msg. delay	Known Δ		No known Δ
Permissionless			

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous		Sluggish	Partially/Asynchronous
Honest msg. delay	Known Δ	Prompt	Known Δ	No known Δ
Permissionless				

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Mobile Sluggish		Partially/Asynchronous
Honest msg. delay	Known Δ	Prompt Sluggish	Known Δ No known Δ	No known Δ
Permissionless				

Problem Statement [GPS19, AMN+20, PS17]

Δ upper bound on the honest message delay

Model	Synchronous	Mobile Sluggish		Partially/Asynchronous
Honest msg. delay	Known Δ	Prompt Sluggish	Known Δ No known Δ	No known Δ
Permissionless				

Is it possible to achieve consensus in the permissionless setting in the presence of mobile sluggish faults?

Overview

- Based on Nakamoto consensus
- All messages are time-lock encrypted
- Set hiding time, $\mathbf{T} = \Delta$

Overview

- Based on Nakamoto consensus
- All messages are time-lock encrypted
- Set hiding time, $\mathbf{T} = \Delta$
- Non-block winners send *decoys*
- Decoys give “cover” for the block winner
- Adversary has to corrupt or deliver messages randomly

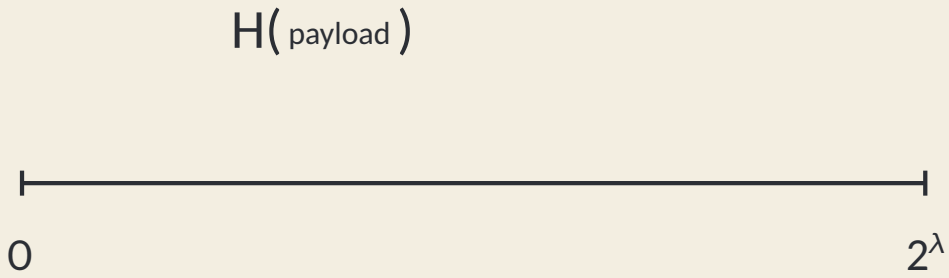
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack

payload

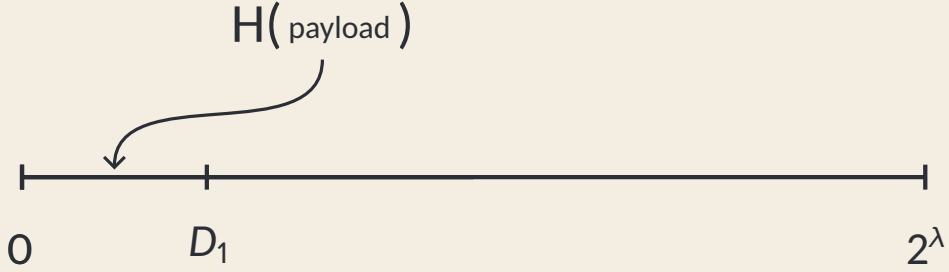
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



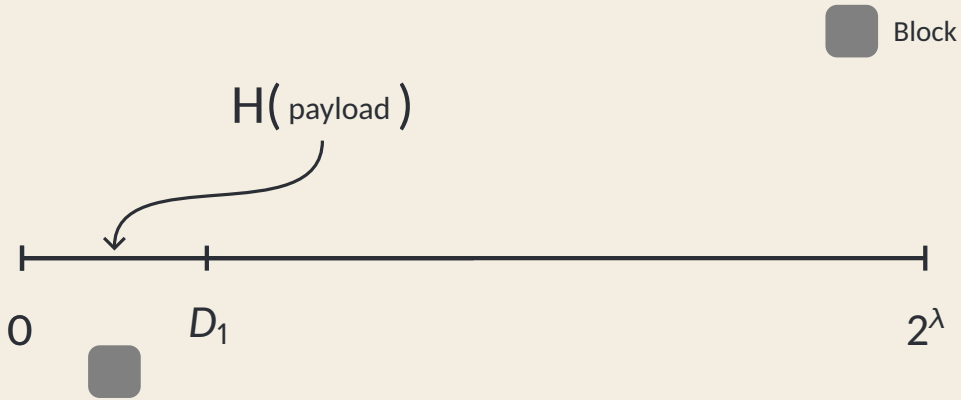
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



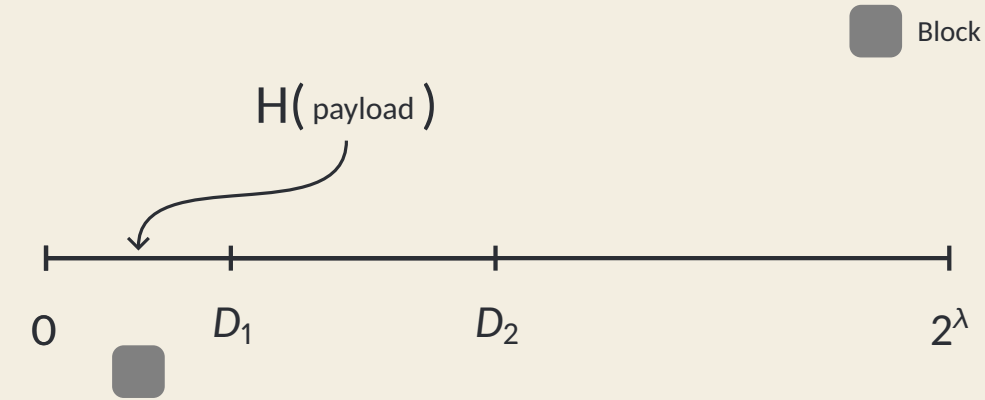
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



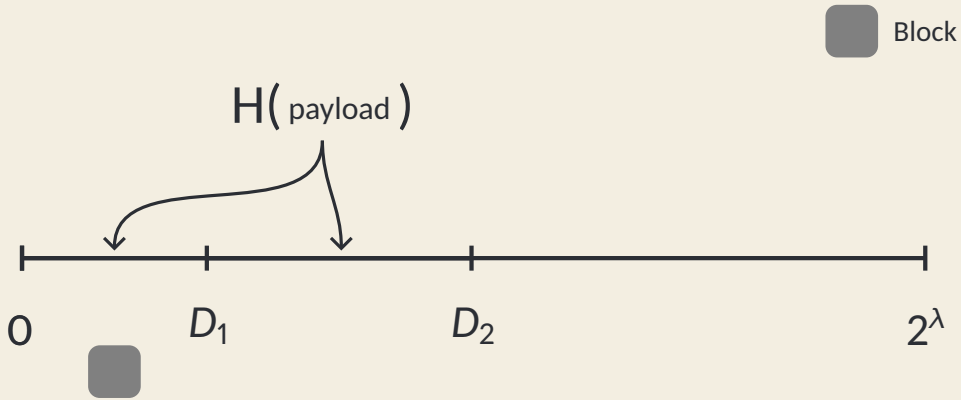
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



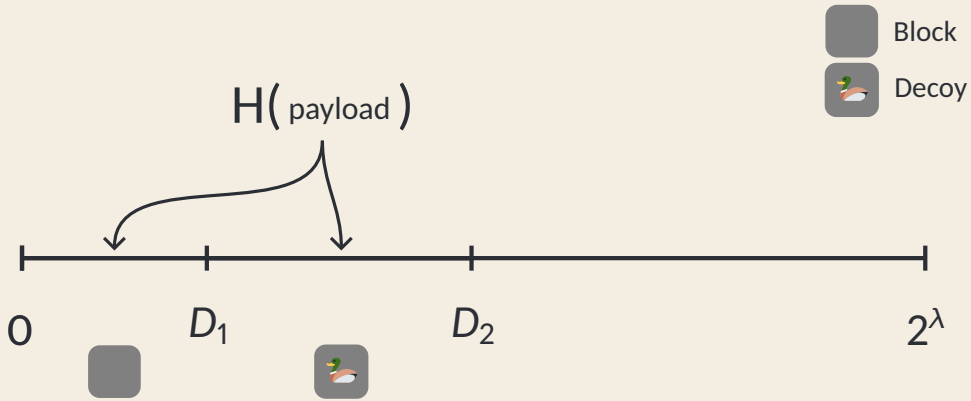
Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



Decoys

- *Decoys*: Dummy TLP messages
- Need to prevent Sybil attack



Protocol

1. Mine Phase:
   - Pick the longest chain
   - Mine a block or a decoy (if possible)
   - Time-lock encrypt the block or decoy

Protocol

1. Mine Phase:
   - Pick the longest chain
   - Mine a block or a decoy (if possible)
   - Time-lock encrypt the block or decoy
2. Solve Phase:
   - Multicast the puzzle
   - Receive puzzles
   - Batch open puzzles
   - Extend the chain and go to step 1

Block Decoy

Protocol

Miner 1

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Mine Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Mine Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Mine Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Mine Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

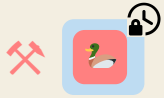
Mine Phase

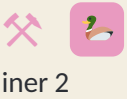
Miner 3

Miner 2

Miner 4

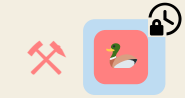
Protocol



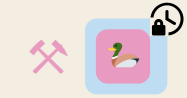


Block Decoy

Protocol

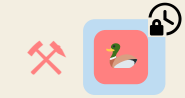


Mine Phase

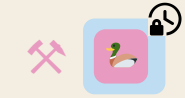


Block Decoy

Protocol

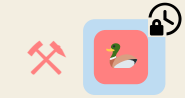


Mine Phase

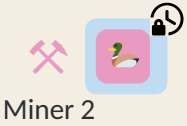


Block Decoy

Protocol

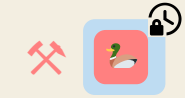


Mine Phase



Block Decoy

Protocol



Miner 1

Solve Phase

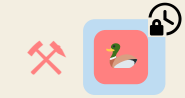
Miner 3

Miner 2

Miner 4

Block Decoy

Protocol



Miner 1

Solve Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Solve Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Solve Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Solve Phase

Block Decoy

Protocol

Solve Phase

Block Decoy

Protocol

Solve Phase

Block Decoy

Protocol

Solve Phase

Block Decoy

Protocol

Solve Phase

Protocol

Miner 1

Solve Phase

Miner 3

Miner 2

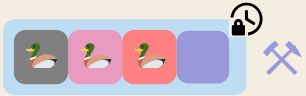
Miner 4

Block Decoy

Protocol

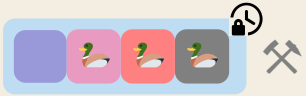
Miner 1

Solve Phase



Miner 3

Miner 2



Miner 4

Block Decoy

Protocol

Miner 1

Solve Phase

Miner 3

Miner 2

Miner 4

Block Decoy

Protocol

Miner 1

Solve Phase

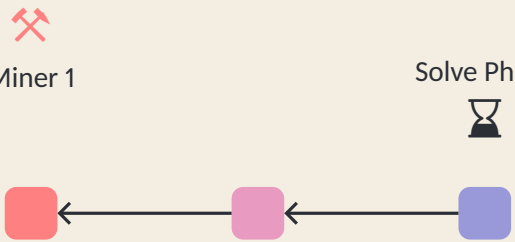
Miner 3

Miner 2

Miner 4

Block Decoy

Protocol



Block Decoy

Protocol

Miner 1

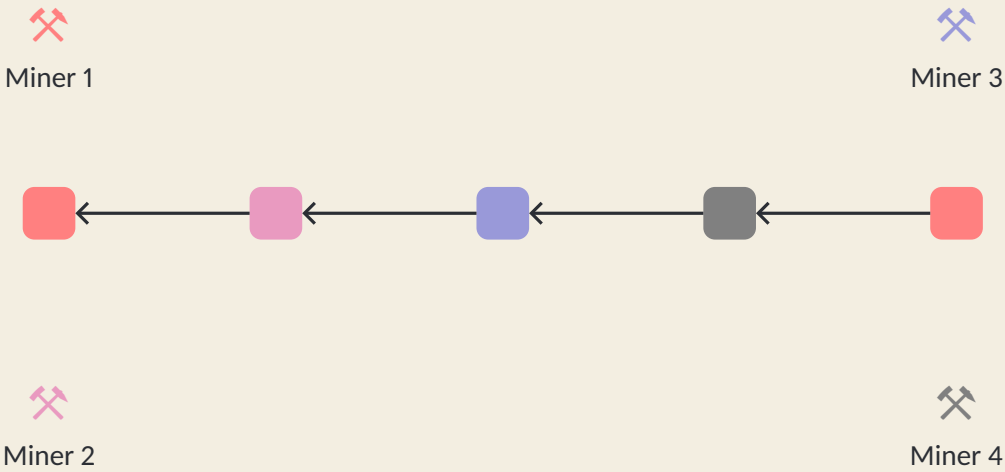
Miner 3

Miner 2

Miner 4

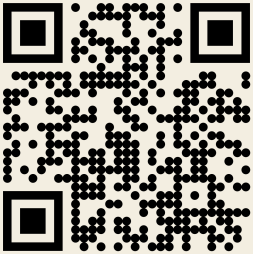
Block Decoy

Protocol



Summary

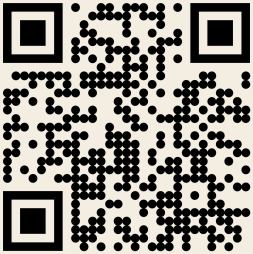
- Transparent batchable TLP with compact puzzle size and unbounded batching



ePrint 2022/1421

Summary

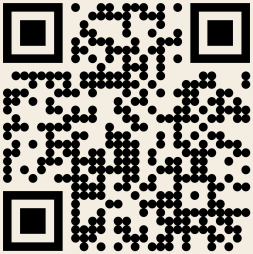
- Transparent batchable TLP with compact puzzle size and unbounded batching
- First permissionless protocol in mobile sluggish model



ePrint 2022/1421

Summary

- Transparent batchable TLP with compact puzzle size and unbounded batching
- First permissionless protocol in mobile sluggish model
- Attack on Nakamoto consensus in mobile sluggish model



ePrint 2022/1421

Summary

- Transparent batchable TLP with compact puzzle size and unbounded batching
- First permissionless protocol in mobile sluggish model
- Attack on Nakamoto consensus in mobile sluggish model
- Using **any** batchable TLP:
  - A generic compiler to convert any weakly adaptive broadcast into a strongly adaptive

ePrint 2022/1421

Summary

- Transparent batchable TLP with compact puzzle size and unbounded batching
- First permissionless protocol in mobile sluggish model
- Attack on Nakamoto consensus in mobile sluggish model
- Using **any** batchable TLP:
  - A generic compiler to convert any weakly adaptive broadcast into a strongly adaptive
  - First expected $O(1)$-round Byzantine broadcast under strongly adaptive and corrupt majority setting

ePrint 2022/1421

Summary

- Transparent batchable TLP with compact puzzle size and unbounded batching
- First permissionless protocol in mobile sluggish model
- Attack on Nakamoto consensus in mobile sluggish model
- Using **any** batchable TLP:
  - A generic compiler to convert any weakly adaptive broadcast into a strongly adaptive
  - First expected $O(1)$-round Byzantine broadcast under strongly adaptive and corrupt majority setting

@s_shravan

ePrint 2022/1421

Broadcast under strongly adaptive and majority corruptions

- Byzantine broadcast is a classical problem in consensus
- Building block for other flavours of consensus
- *Strongly* adaptive adversary:
  - Can corrupt the victim on-the-fly
  - Can perform *after-the-fact* removal in the ongoing round
- *Weakly* adaptive adversary:
  - Can corrupt the victim on-the-fly
  - Cannot perform *after-the-fact* removal

References I

[AMN+20] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin.
Sync HotStuff: Simple and Practical Synchronous State Machine Replication.
In *2020 IEEE Symposium on Security and Privacy (SP)*, 2020.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan.
Constrained key-homomorphic prfs from standard lattice assumptions.
In *Theory of Cryptography Conference*, pages 1–30, 2015.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos.
The Bitcoin Backbone Protocol: Analysis and Applications.
In *Advances in Cryptology - EUROCRYPT 2015*, 2015.

References II

[GPS19] Yue Guo, Rafael Pass, and Elaine Shi.
Synchronous, with a Chance of Partition Tolerance.
In *Advances in Cryptology – CRYPTO 2019*, 2019.

[LG19] Jing Li and Dongning Guo.
On Analysis of the Bitcoin and Prism Backbone Protocols, 2019.

[May92] Timothy May.
Timed-release crypto.
1992.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan.
Homomorphic Time-Lock Puzzles and Applications.
In *Advances in Cryptology – CRYPTO 2019*, 2019.

References III

[PS17] Rafael Pass and Elaine Shi.
Rethinking Large-Scale Consensus.
In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat.
Analysis of the Blockchain Protocol in Asynchronous Networks.
In *Advances in Cryptology – EUROCRYPT 2017*, 2017.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner.
Time-Lock Puzzles and Timed-Release Crypto.
Technical report, 1996.

References IV

[TBM+20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder.

Verifiable timed signatures made practical.

In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabian Laguillaumie, and Giulio Malavolta.

Efficient cca timed commitments in class groups.

In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[tez22] Time-lock: Block producer extractable value - tezos, 2022.

[Online; accessed 01-Sept-2022].

References V

[WXDS20] Jun Wan, Hanshen Xiao, Srinivas Devadas, and Elaine Shi.

Round-Efficient Byzantine Broadcast Under Strongly Adaptive and Majority Corruptions.

In *Theory of Cryptography*, 2020.