Permissionless Clock Synchronization with Public Setup

https://ia.cr/2022/1220

TCC 2022

[1]Texas A&M University [2]University of Edinburgh [3]IOG

Juan Garay[1], Aggelos Kiayias[2,3], Yu Shen[2]

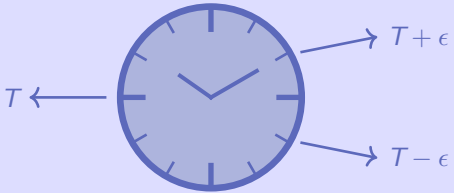
Clocks in Distributed Systems

A set of n processors (t might be corrupted), with physical clocks within a linear envelope[1] of real time, should realize logical clocks that satisfy clock synchronization conditions.

[1] A function $f : \mathbb{R} \to \mathbb{R}$ is within a (U, L)-linear envelope if and only if it holds that $L \cdot x \leq f(x) \leq U \cdot x$, for all x.

Clocks in Distributed Systems

A set of n processors (t might be corrupted), with physical clocks within a linear envelope[1] of real time, should realize logical clocks that satisfy clock synchronization conditions.



Precision

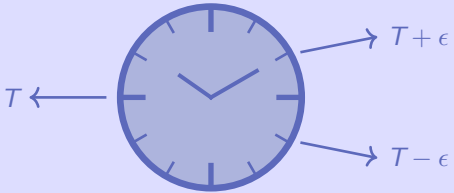
Honest parties maintain close logical clocks.

(Bounded skew)

[1] A function $f : \mathbb{R} \to \mathbb{R}$ is within a (U, L)-linear envelope if and only if it holds that $L \cdot x \leq f(x) \leq U \cdot x$, for all x.

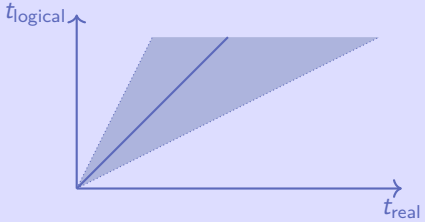
Clocks in Distributed Systems

A set of n processors (t might be corrupted), with physical clocks within a linear envelope[1] of real time, should realize logical clocks that satisfy clock synchronization conditions.



Precision

Honest parties maintain close logical clocks.
(Bounded skew)



Accuracy

Honest parties report logical time within a linear envelope of the real time.

[1] A function $f : \mathbb{R} \to \mathbb{R}$ is within a (U, L)-linear envelope if and only if it holds that $L \cdot x \leq f(x) \leq U \cdot x$, for all x.

Classical Clock Synchronization Protocols

This problem has been studied for over 40 years [Lam78; LM84; Hal+84; DHS86; ST87; WL88; Abr+19; LL22].

[Lam78] Leslie Lamport. **"Time, Clocks, and the Ordering of Events in a Distributed System".**

[LM84] Leslie Lamport and P. M. Melliar-Smith. **"Byzantine Clock Synchronization".**

[Hal+84] Joseph Y. Halpern, Barbara Simons, Ray Strong, and Danny Dolev. **"Fault-Tolerant Clock Synchronization".**

[DHS86] Danny Dolev, Joseph Y. Halpern, and H. Raymond Strong. **"On the Possibility and Impossibility of Achieving Clock Synchronization".**

[ST87] T. K. Srikanth and Sam Toueg. **"Optimal clock synchronization".**

[WL88] Jennifer Lundelius Welch and Nancy Lynch. **"A new fault-tolerant algorithm for clock synchronization".**

[Abr+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. **"Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience".**

[LL22] Christoph Lenzen and Julian Loss. **"Optimal Clock Synchronization with Signatures".**

Classical Clock Synchronization Protocols

This problem has been studied for over 40 years [Lam78; LM84; Hal+84; DHS86; ST87; WL88; Abr+19; LL22].

- ✓ Optimal skew: $\approx$ transmission uncertainty when # of parties is large.
- ✓ Optimal linear envelope: as good as the linear envelope on physical clocks.

[Lam78] Leslie Lamport. **"Time, Clocks, and the Ordering of Events in a Distributed System".**

[LM84] Leslie Lamport and P. M. Melliar-Smith. **"Byzantine Clock Synchronization".**

[Hal+84] Joseph Y. Halpern, Barbara Simons, Ray Strong, and Danny Dolev. **"Fault-Tolerant Clock Synchronization".**

[DHS86] Danny Dolev, Joseph Y. Halpern, and H. Raymond Strong. **"On the Possibility and Impossibility of Achieving Clock Synchronization".**

[ST87] T. K. Srikanth and Sam Toueg. **"Optimal clock synchronization".**

[WL88] Jennifer Lundelius Welch and Nancy Lynch. **"A new fault-tolerant algorithm for clock synchronization".**

[Abr+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. **"Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience".**

[LL22] Christoph Lenzen and Julian Loss. **"Optimal Clock Synchronization with Signatures".**

Classical Clock Synchronization Protocols

This problem has been studied for over 40 years [Lam78; LM84; Hal+84; DHS86; ST87; WL88; Abr+19; LL22].

- ✓ Optimal skew: $\approx$ transmission uncertainty when # of parties is large.
- ✓ Optimal linear envelope: as good as the linear envelope on physical clocks.
- ● "Permissioned" environment.

[Lam78] Leslie Lamport. **"Time, Clocks, and the Ordering of Events in a Distributed System".**

[LM84] Leslie Lamport and P. M. Melliar-Smith. **"Byzantine Clock Synchronization".**

[Hal+84] Joseph Y. Halpern, Barbara Simons, Ray Strong, and Danny Dolev. **"Fault-Tolerant Clock Synchronization".**

[DHS86] Danny Dolev, Joseph Y. Halpern, and H. Raymond Strong. **"On the Possibility and Impossibility of Achieving Clock Synchronization".**

[ST87] T. K. Srikanth and Sam Toueg. **"Optimal clock synchronization".**

[WL88] Jennifer Lundelius Welch and Nancy Lynch. **"A new fault-tolerant algorithm for clock synchronization".**

[Abr+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. **"Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience".**

[LL22] Christoph Lenzen and Julian Loss. **"Optimal Clock Synchronization with Signatures".**

Permissionless Protocols

These classical synchronization protocols are challenged by a permissionless environment.

Permissionless Protocols

These classical synchronization protocols are challenged by a permissionless environment.

Model	Prior	Permissionless
# of parties	Fixed	Dynamic
Online assumption	Always active	Unannounced disappearance
Setup	PKI	CRS, Proof-of-Work; Proof-of-Stake
Joining process	Upon approval from sufficiently many participants	Passive bootstrapping. (No need for approval)

Permissionless Protocols

These classical synchronization protocols are challenged by a permissionless environment.

Model	Prior	Permissionless
# of parties	Fixed	Dynamic
Online assumption	Always active	Unannounced disappearance
Setup	PKI	CRS, Proof-of-Work; Proof-of-Stake
Joining process	Upon approval from sufficiently many participants	Passive bootstrapping. (No need for approval)

- Permissionless protocols can no longer use n and t in the protocol logic.

Permissionless Protocols

These classical synchronization protocols are challenged by a permissionless environment.

Model	Prior	Permissionless
# of parties	Fixed	Dynamic
Online assumption	Always active	Unannounced disappearance
Setup	PKI	CRS, Proof-of-Work; Proof-of-Stake
Joining process	Upon approval from sufficiently many participants	Passive bootstrapping. (No need for approval)

- Permissionless protocols can no longer use n and t in the protocol logic.

→ Almost all existing protocols will not work (cf. Ouroboros Chronos [Bad+21, Eurocrypt '21]).

Clocks in Bitcoin

Bitcoin miners adjust their time based on three different sources[2]:

- Local system clock
- Median peer clocks
- Human operator

[2] If the first two sources disagree, the program will ask for the third.

Clocks in Bitcoin

Bitcoin miners adjust their time based on three different sources[2]:

Local system clock | Median peer clocks | Human operator

- System clocks are usually adjusted via NTP. (single point failure)

[2] If the first two sources disagree, the program will ask for the third.

Clocks in Bitcoin

Bitcoin miners adjust their time based on three different sources[2]:

Local system clock | Median peer clocks | Human operator

- System clocks are usually adjusted via NTP. (single point failure)
- In Bitcoin protocol, some amount of deviation (70min $\sim$ 2h) is allowed.

[2] If the first two sources disagree, the program will ask for the third.

Clocks in Bitcoin

Bitcoin miners adjust their time based on three different sources[2]:

Local system clock | Median peer clocks | Human operator

- System clocks are usually adjusted via NTP. (single point failure)
- In Bitcoin protocol, some amount of deviation (70min $\sim$ 2h) is allowed.
- Many blocks in Bitcoin report timestamp smaller than its parent.

[2] If the first two sources disagree, the program will ask for the third.

Clocks in Bitcoin

Bitcoin miners adjust their time based on three different sources[2]:

Local system clock | Median peer clocks | Human operator

- System clocks are usually adjusted via NTP. (single point failure)
- In Bitcoin protocol, some amount of deviation (70min $\sim$ 2h) is allowed.
- Many blocks in Bitcoin report timestamp smaller than its parent.

$\rightarrow$ Clocks in Bitcoin network are not synchronized.

[2] If the first two sources disagree, the program will ask for the third.

Related Work

- Having access to synchronized clocks is an essential feature in Nakamoto Consensus, which is reflected in previous analyses [GKL15; PSS17; GKL17; Bad+17].

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications".**

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. **"Analysis of the Blockchain Protocol in Asynchronous Networks".**

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol with Chains of Variable Difficulty".**

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. **"Bitcoin as a Transaction Ledger: A Composable Treatment".**

Related Work

- Having access to synchronized clocks is an essential feature in Nakamoto Consensus, which is reflected in previous analyses [GKL15; PSS17; GKL17; Bad+17].
- ✓ **Ouroboros Chronos** [Bad+21] is the first permissionless **Proof-of-Stake** clock synchronization protocol.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications".**

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. **"Analysis of the Blockchain Protocol in Asynchronous Networks".**

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol with Chains of Variable Difficulty".**

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. **"Bitcoin as a Transaction Ledger: A Composable Treatment".**

[Bad+21] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. **"Dynamic Ad Hoc Clock Synchronization".**

Related Work

- Having access to synchronized clocks is an essential feature in Nakamoto Consensus, which is reflected in previous analyses [GKL15; PSS17; GKL17; Bad+17].
- ✓ **Ouroboros Chronos** [Bad+21] is the first permissionless **Proof-of-Stake** clock synchronization protocol.
- ✗ The construction from Ouroboros Chronos, when applied in **Proof-of-Work** context, will eventually become insecure when there is a steady increase (or decrease) of participants.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications".**

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. **"Analysis of the Blockchain Protocol in Asynchronous Networks".**

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol with Chains of Variable Difficulty".**

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. **"Bitcoin as a Transaction Ledger: A Composable Treatment".**

[Bad+21] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. **"Dynamic Ad Hoc Clock Synchronization".**

Our Results

Is it possible for a dynamically changing population of peers to synchronize their clocks utilizing only a public setup and assuming PoW?

Is there a blockchain protocol in the PoW setting that has no dependency on a publicly accessible global clock?

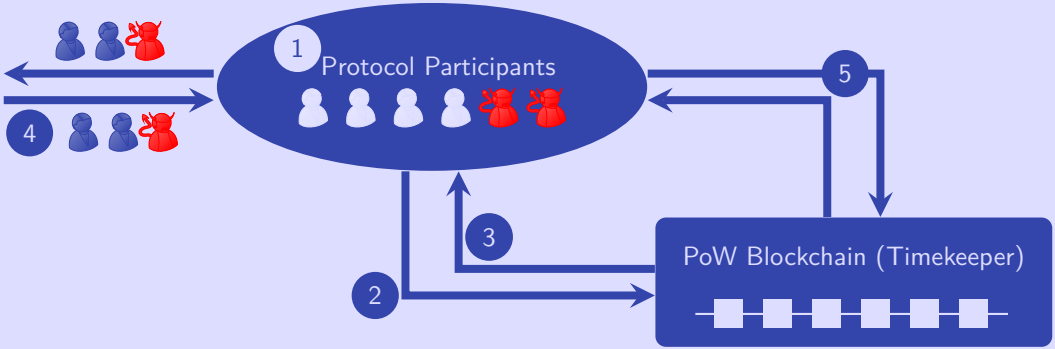
Our Results

Is it possible for a dynamically changing population of peers to synchronize their clocks utilizing only a public setup and assuming PoW?

Is there a blockchain protocol in the PoW setting that has no dependency on a publicly accessible global clock?

Main Theorem (informal). A new PoW-based protocol (Timekeeper) solves the permissionless clock synchronization problem assuming **bounded dynamic participation** and **honest majority** in terms of random oracle queries.

Informal Solution



1. Parties use **2-for-1 Proof-of-Work** to mine and diffuse sync-messages containing local time.

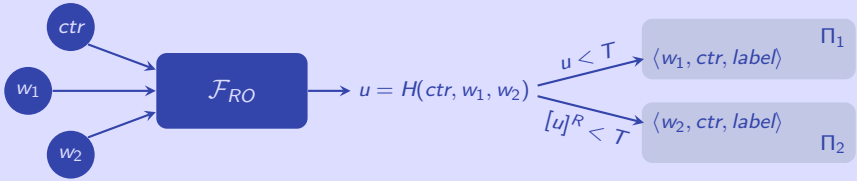
2-for-1 Proof-of-Work

2-for-1 PoW [GKL15, Eurocrypt '15] is a primitive that composes two or more PoW mining processes with access to a single oracle $H(\cdot)$.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications".**

2-for-1 Proof-of-Work

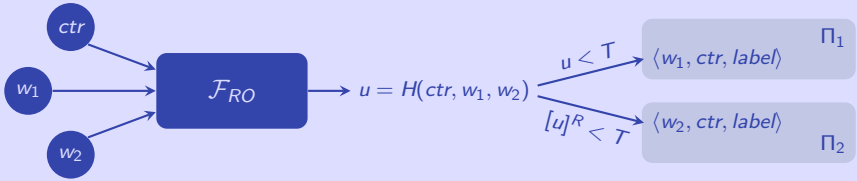
2-for-1 PoW [GKL15, Eurocrypt '15] is a primitive that composes two or more PoW mining processes with access to a single oracle $H(\cdot)$.



[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications"**.

2-for-1 Proof-of-Work

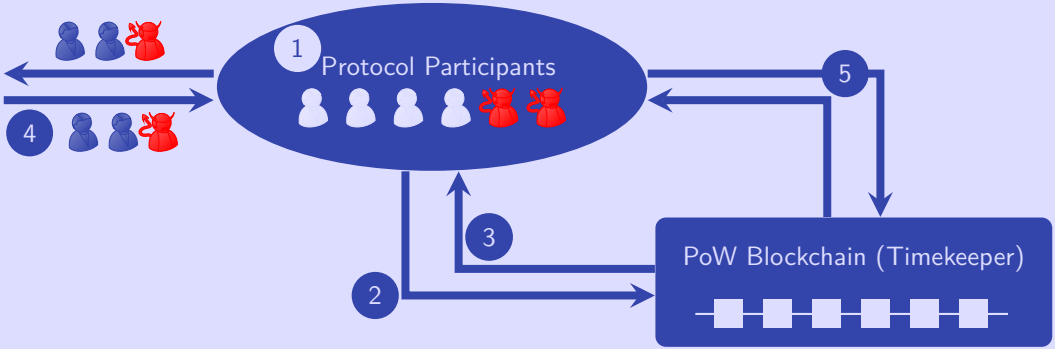
2-for-1 PoW [GKL15, Eurocrypt '15] is a primitive that composes two or more PoW mining processes with access to a single oracle $H(\cdot)$.



Applications: PoW-based BA for honest majority [GKL15]; parallel chains [Bag+19; Fit+20]; fair reward sharing [PS17].

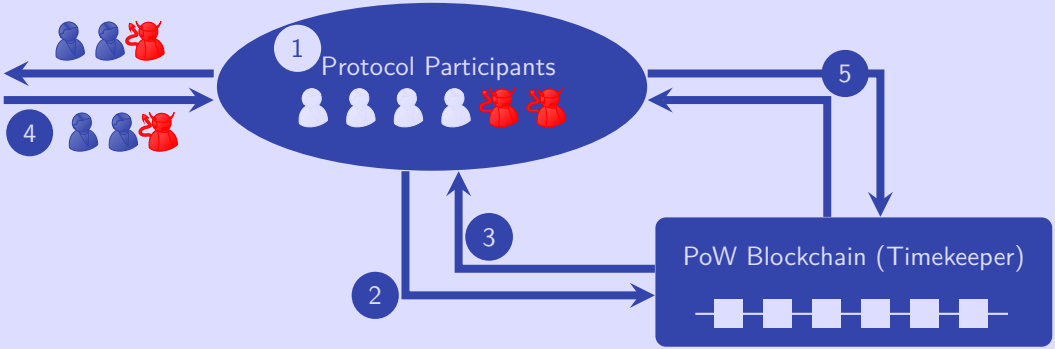
[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. **"The Bitcoin Backbone Protocol: Analysis and Applications"**.

Informal Solution



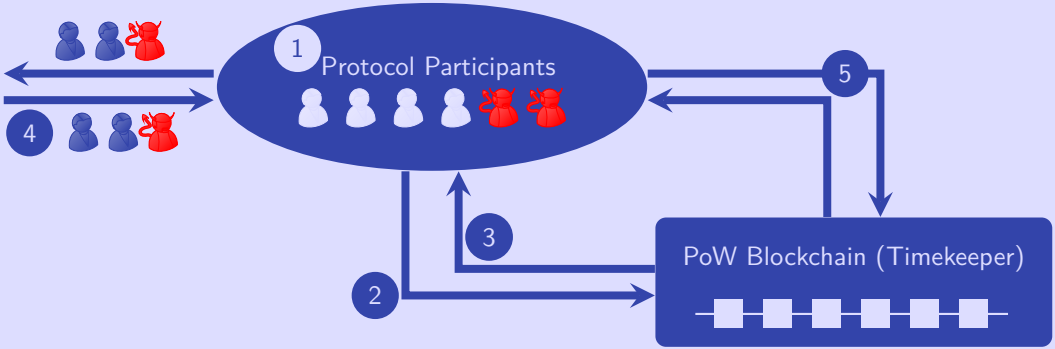
2. Reach consensus over the set of sync-messages using blockchain periodically.

Informal Solution



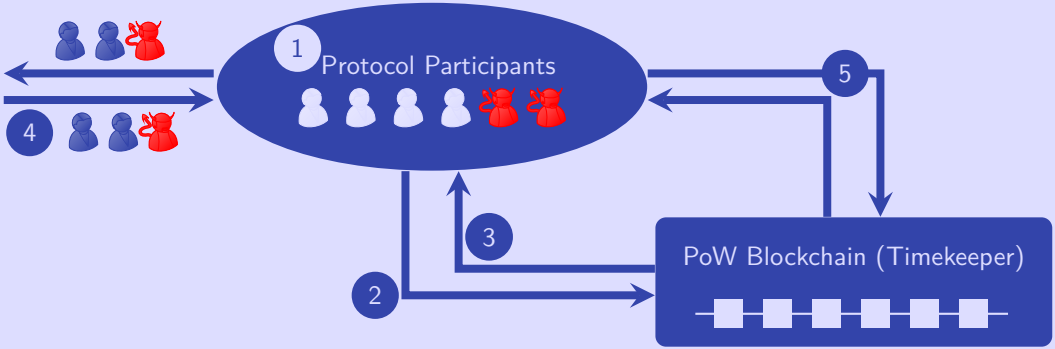
3 Local adjustment — median of "delay" (comparing timestamp recorded with arrival time).

Informal Solution



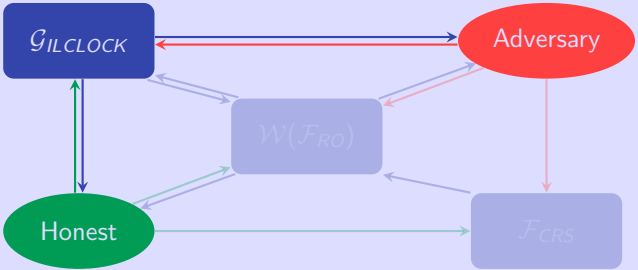
4 Newly joint parties can sync with honest parties by passively observing recent adjustments.

Informal Solution



5 Blockchain can react to party fluctuation by a novel recalculation mechanism.

Imperfect Clocks



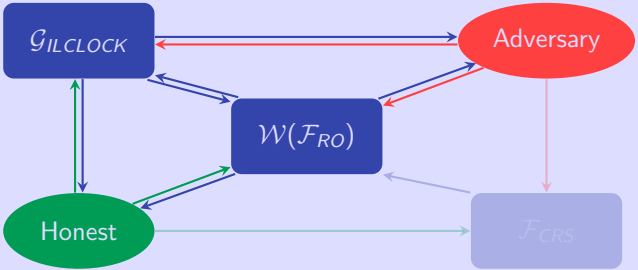
- Imperfect clock forwards "nominal time" after all honest clocks have "ping" it [Kat+13].
- Adversary can set additional drifts to honest clocks [Bad+21].

Honest parties cannot learn the exact time, but can proceed at "roughly" the same speed.

[Kat+13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. **"Universally Composable Synchronous Computation"**.

[Bad+21] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. **"Dynamic Ad Hoc Clock Synchronization"**.

Random Oracle and its Wrapper



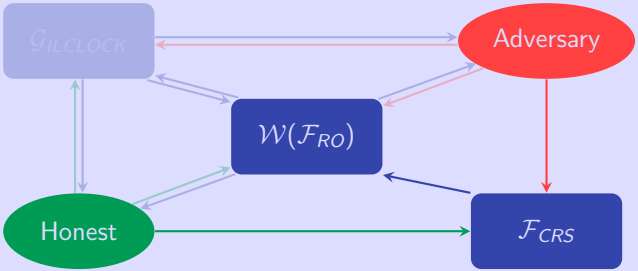
- Random oracle wrapper limits the adversary's power to make queries [Bad+17; Gar+20].

We define **honest majority** with respect to the random oracle queries in terms of the nominal time.

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. **"Bitcoin as a Transaction Ledger: A Composable Treatment"**.

[Gar+20] Juan Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. **"Resource-Restricted Cryptography: Revisiting MPC Bounds in the Proof-of-Work Era"**.

Common Reference String



- Adversary has unlimited (but polynominally bounded) power to make RO queries before CRS is retrieved.

We model the public setup via common reference string.

1. Introduction

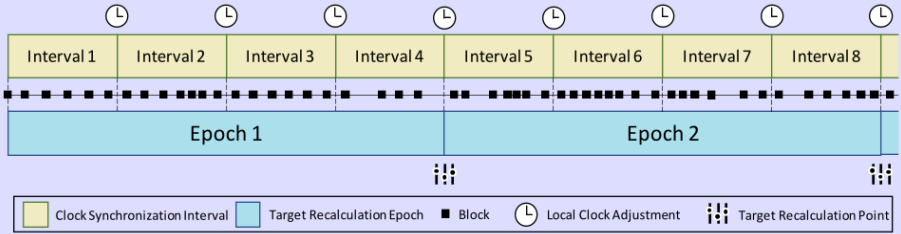
2. Informal Solution

3. Model

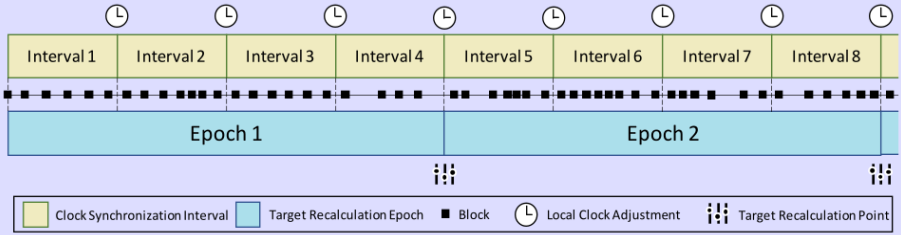
4. Protocol Details

5. Security Analysis

Timekeeper Protocol

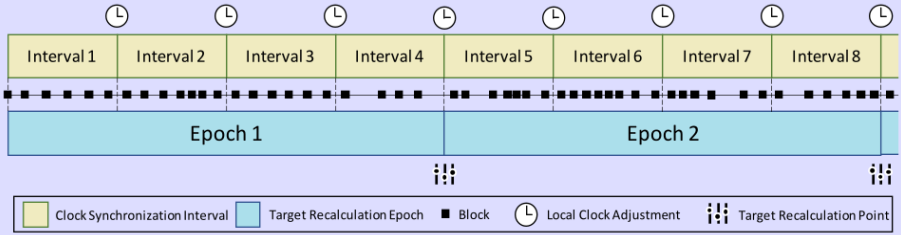


Timekeeper Protocol



- All events are triggered by parties' local clocks.

Timekeeper Protocol



- All events are triggered by parties' local clocks.
- Parties use 2-for-1 PoW to mine and emit synchronization beacons to sync their local clocks.

A New Timestamp Scheme

We extend the timestamp scheme from $\langle time \rangle$ to $\langle interval, time \rangle$.

A New Timestamp Scheme

We extend the timestamp scheme from $\langle time \rangle$ to $\langle interval, time \rangle$.

- Chain should include monotonically increasing timestamps.

A New Timestamp Scheme

We extend the timestamp scheme from $\langle time \rangle$ to $\langle interval, time \rangle$.

- Chain should include monotonically increasing timestamps.
- Recalibration may "rewind" local clocks.

A New Timestamp Scheme

We extend the timestamp scheme from $\langle time \rangle$ to $\langle interval, time \rangle$.

- Chain should include monotonically increasing timestamps.
- Recalibration may "rewind" local clocks.
- Easy to distinguish the epoch of blocks mined at the boundary.

A New Timestamp Scheme

We extend the timestamp scheme from $\langle time \rangle$ to $\langle interval, time \rangle$.

- Chain should include monotonically increasing timestamps.
- Recalibration may "rewind" local clocks.
- Easy to distinguish the epoch of blocks mined at the boundary.
- After setting clocks backwards, it is unreasonable to let honest parties stop mining.

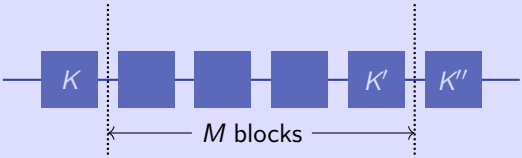
A New Target Recalculation Function

We reverse Bitcoin's target recalculation function.

A New Target Recalculation Function

We reverse Bitcoin's target recalculation function.

Bitcoin's Function



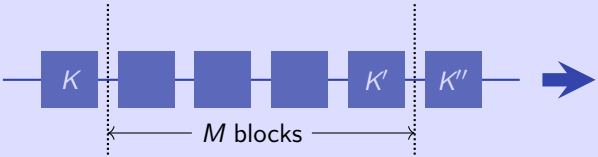
$$Target(K'') = Target(K) * \frac{K'.timestamp - K.timestamp}{\Lambda}.$$

Λ : ideal epoch duration (2 weeks in use)

A New Target Recalculation Function

We reverse Bitcoin's target recalculation function.

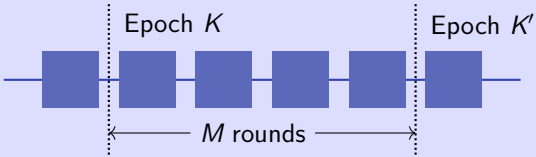
Bitcoin's Function



$$Target(K'') = Target(K) * \frac{K'.timestamp - K.timestamp}{\Lambda}.$$

Λ : ideal epoch duration (2 weeks in use)

Our New Function



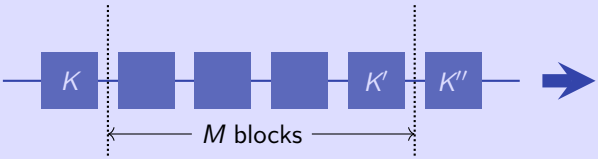
$$Target(K') = Target(K) * \frac{\Lambda}{(\#\text{ of blocks in } K)}.$$

Λ : ideal # of blocks in an epoch

A New Target Recalculation Function

We reverse Bitcoin's target recalculation function.

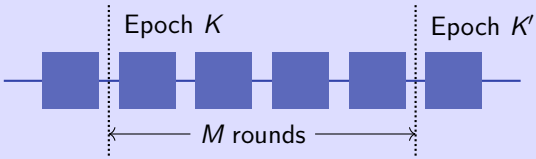
Bitcoin's Function



$$Target(K'') = Target(K) * \frac{K'.timestamp - K.timestamp}{\Lambda}.$$

Λ : ideal epoch duration (2 weeks in use)

Our New Function



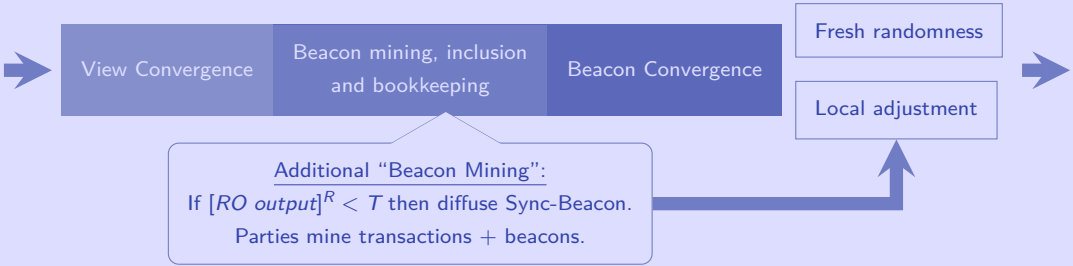
$$Target(K') = Target(K) * \frac{\Lambda}{(\#\text{ of blocks in } K)}.$$

Λ : ideal # of blocks in an epoch

No timestamp is needed in the new function!

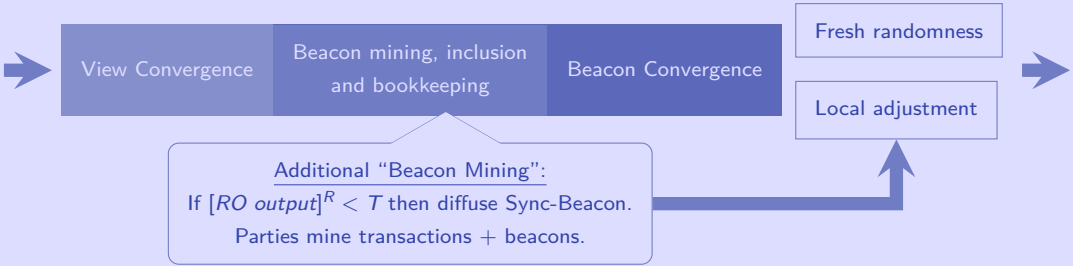
Synchronization Intervals

A synchronization interval consists of 3 phases.



Synchronization Intervals

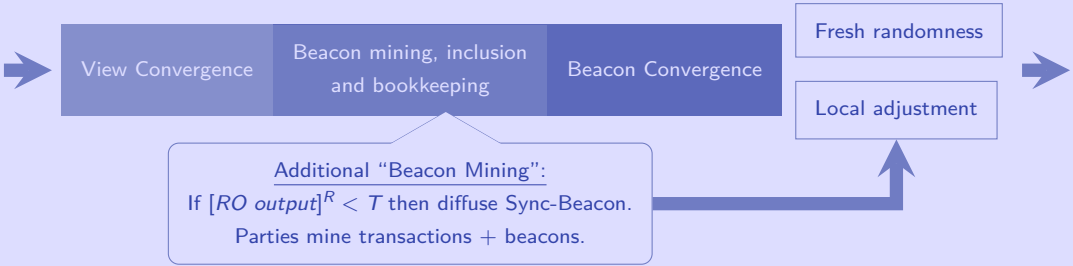
A synchronization interval consists of 3 phases.



1. **View convergence:** Wait for a consistent view of the last interval.

Synchronization Intervals

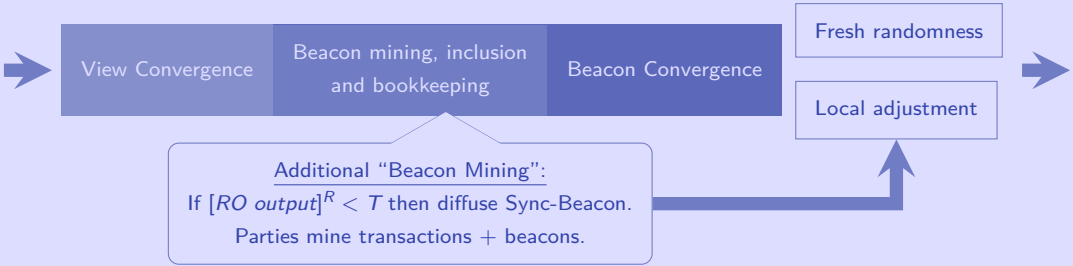
A synchronization interval consists of 3 phases.



1. **View convergence:** Wait for a consistent view of the last interval.
2. **Mining, inclusion and bookkeeping:** Use **2-for-1 PoW** to mine and diffuse beacons – a tiny block that contains local time; honest parties bookkeep the arrival time of beacons.

Synchronization Intervals

A synchronization interval consists of 3 phases.



1. **View convergence:** Wait for a consistent view of the last interval.
2. **Mining, inclusion and bookkeeping:** Use **2-for-1 PoW** to mine and diffuse beacons – a tiny block that contains local time; honest parties bookkeep the arrival time of beacons.
3. **Beacon Convergence:** Wait for a consistent view of the beacons in current interval.

Local Adjustment

Parties adjust their clocks locally. They first compute the "delay" of each beacon based on its local arrival time and recorded timestamp, then they add the **median** of these delays to local time.

Local Adjustment

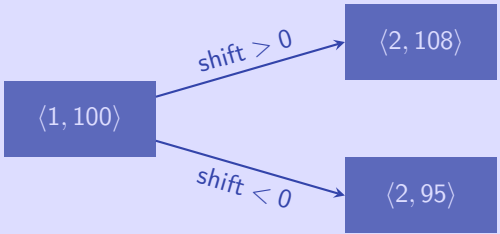
Parties adjust their clocks locally. They first compute the "delay" of each beacon based on its local arrival time and recorded timestamp, then they add the **median** of these delays to local time.

$$\text{shift} = \text{median}\{\texttt{SB}.\mathit{timestamp} - \texttt{SB}.\mathit{arrivalTime} \mid \texttt{SB} \text{ in beacon set}\}$$

Local Adjustment

Parties adjust their clocks locally. They first compute the "delay" of each beacon based on its local arrival time and recorded timestamp, then they add the **median** of these delays to local time.

$$\text{shift} = \text{median}\{\texttt{SB}.\mathit{timestamp} - \texttt{SB}.\mathit{arrivalTime} \mid \texttt{SB} \text{ in beacon set}\}$$



Skip some rounds in the next interval.

Set clock backward and continue with the rewound timestamp.

1. Introduction

2. Informal Solution

3. Model

4. Protocol Details

5. Security Analysis

Proof Roadmap

Assuming a safe startup, we present an inductive-style proof.

- If at the onset, the PoW difficulty is appropriately set and the steady block-generation rate lasts during the whole clock synchronization interval, parties can maintain good skews after they enter the next interval and the shift value they compute to adjust their clocks is properly bounded.
- If good skews and certain time adjustment calculations are maintained during a target recalculation epoch, the block production rate will be properly controlled in the next epoch.

Thank You

References

[Abr+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. "Synchronous Byzantine Agreement with Expected O(1) Rounds, Expected $O(n^2)$ Communication, and Optimal Resilience". In: **Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers**. Ed. by Ian Goldberg and Tyler Moore. Vol. 11598. Lecture Notes in Computer Science. Springer, 2019, pp. 320–334.

[Bad+21] Christian Badertscher, Peter Gaži, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. "Dynamic Ad Hoc Clock Synchronization". In: **Advances in Cryptology – EUROCRYPT 2021**. Ed. by Anne Canteaut and François-Xavier Standaert. Cham: Springer International Publishing, 2021, pp. 399–428.

[Bad+17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. "Bitcoin as a Transaction Ledger: A Composable Treatment". In: **Advances in Cryptology – CRYPTO 2017**. Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer International Publishing, 2017, pp. 324–356.

References

[Bag+19] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. "Prism: Deconstructing the Blockchain to Approach Physical Limits". In: **Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security**. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 585–602.

[DHS86] Danny Dolev, Joseph Y. Halpern, and H. Raymond Strong. "On the Possibility and Impossibility of Achieving Clock Synchronization". In: **J. Comput. Syst. Sci.** 32.2 (1986), pp. 230–250.

[Fit+20] Matthias Fitzi, Peter Gaži, Aggelos Kiayias, and Alexander Russell. "Ledger Combiners for Fast Settlement". In: **Theory of Cryptography**. Ed. by Rafael Pass and Krzysztof Pietrzak. Cham: Springer International Publishing, 2020, pp. 322–352.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol: Analysis and Applications". In: **Advances in Cryptology - EUROCRYPT 2015**. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 281–310.

References

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. "The Bitcoin Backbone Protocol with Chains of Variable Difficulty". In: **Advances in Cryptology – CRYPTO 2017**. Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer International Publishing, 2017, pp. 291–323.

[Gar+20] Juan Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. "Resource-Restricted Cryptography: Revisiting MPC Bounds in the Proof-of-Work Era". In: **Advances in Cryptology – EUROCRYPT 2020**. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 129–158.

[Hal+84] Joseph Y. Halpern, Barbara Simons, Ray Strong, and Danny Dolev. "Fault-Tolerant Clock Synchronization". In: **Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing**. PODC '84. Vancouver, British Columbia, Canada: Association for Computing Machinery, 1984, pp. 89–102.

References

[Kat+13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. "Universally Composable Synchronous Computation". In: **Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography**. TCC'13. Tokyo, Japan: Springer-Verlag, 2013, pp. 477–498.

[Lam78] Leslie Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". In: **Commun. ACM** 21.7 (1978), pp. 558–565.

[LM84] Leslie Lamport and P. M. Melliar-Smith. "Byzantine Clock Synchronization". In: **Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing**. PODC '84. Vancouver, British Columbia, Canada: Association for Computing Machinery, 1984, pp. 68–74.

[LL22] Christoph Lenzen and Julian Loss. "Optimal Clock Synchronization with Signatures". In: **Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing**. PODC'22. Salerno, Italy: Association for Computing Machinery, 2022, pp. 440–449.

References

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. "Analysis of the Blockchain Protocol in Asynchronous Networks". In: **Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II**. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. 2017, pp. 643–673.

[PS17] Rafael Pass and Elaine Shi. "FruitChains: A Fair Blockchain". In: **Proceedings of the ACM Symposium on Principles of Distributed Computing**. PODC '17. Washington, DC, USA: Association for Computing Machinery, 2017, pp. 315–324.

[ST87] T. K. Srikanth and Sam Toueg. "Optimal clock synchronization". In: **J. ACM** 34.3 (1987), pp. 626–645.

[WL88] Jennifer Lundelius Welch and Nancy Lynch. "A new fault-tolerant algorithm for clock synchronization". In: **Information and Computation** 77.1 (1988), pp. 1–36.