On Elapsed Time Consensus Protocols

Mic Bowman[1], Debajyoti Das[2](✉), Avradip Mandal[3], and Hart Montgomery[3]

[1] Intel Labs, Santa Clara, USA
mic.bowman@intel.com
[2] imec-COSIC, KU Leuven, Leuven, Belgium
debajyoti.das@esat.kuleuven.be
[3] Fujitsu Laboratories of America, Sunnyvale, USA
{amandal,hhmontgomery}@fujitsu.com

Abstract. *Proof of Elapsed Time* (PoET) is a Nakamoto-style consensus algorithm where proof of work is replaced by a wait time randomly generated by a trusted execution environment (TEE). PoET was originally developed by Intel engineers and contributed to Hyperledger Sawtooth, but has never been formally defined or analyzed. In particular, PoET enables consensus on a bitcoin-like scale without having to resort to mining. *Proof of Luck* (PoL), designed by Milutinovic et al., is a similar (but not identical) protocol that also builds a Nakamoto-style consensus algorithm using a TEE. Like PoET, it also lacks a formal proof.

In this work, we formally define a simplified version of PoET and PoL, which we call *elapsed time consensus* (ET) with a trusted timer. We prove the security of our ET consensus with a trusted timer given an honest majority assumption in a model that generalizes the bitcoin backbone model proposed by Garay et al. which we call the *elapsed time backbone model.* Our model and protocol aim to capture the essence of PoET and PoL while ignoring some of the more practical difficulties associated with such protocols, such as bootstrapping and setting up the TEE.

The PoET protocol also contains a function called the z-test that limits the number of blocks a player can publish in any particular set of blocks of some (larger) size. Surprisingly, by improving this z-test we can prove the security of our ET consensus protocol *without* any TEEs with a (slightly stronger) honest majority assumption. This implies that Nakamoto-style consensus with rate limiting and no proofs of work can be used to obtain scalable consensus in a permissioned setting: in other words, "bitcoin without proofs of work" can be made secure without a TEE for private blockchains.

1 Introduction

In today's interconnected world, it is important to be able to share data widely but in a selective manner. Efficient distributed databases have been known for quite some time and continue to improve [48]. However, basic distributed

D. Das—This work was started during Debajyoti Das' internship at Fujitsu Laboratories of America and continued during the his PhD studies at Purdue.

© Springer Nature Switzerland AG 2021
A. Adhikari et al. (Eds.): INDOCRYPT 2021, LNCS 13143, pp. 559–583, 2021.
https://doi.org/10.1007/978-3-030-92518-5_25

databases have a core problem: they have absolutely no protection from malicious users. Since we do not live in a perfect world, we cannot expect database users to be angels [38], and this leads to many practical issues when using distributed databases: what happens when two people or entities that do not trust each other need to share data? What if some participants in the database are outright malicious?

Almost two decades ago, Castro and Liskov [18] came up with a clever solution to this problem: the practical byzantine fault tolerant (PBFT) consensus algorithm. PBFT was a clever and practical invention: it allowed people to use what were essentially distributed databases that tolerated up to a third of the users being malicious. This was a big improvement over basic distributed databases, but still did not allow for truly public databases. In addition, PBFT protocols require a large amount of communication between participants–$O\left(n^2\right)$, for n parties [25], which makes them very difficult to scale. So while PBFT protocols proved to be very useful for many applications of distributed computing, they did not fully solve the fundamental problem at hand.

In 2008, another new technology radically changed the state of distributed databases: bitcoin [41]. Someone using the pseudonym Satoshi Nakamoto designed what amounted to a new distributed database with some pretty incredible properties: the database is fully public, so anyone can participate, and (probabilistic) consensus in the optimistic case only requires $O\left(n\right)$ communication, meaning that it is easy for tens of thousands of users to participate in bitcoin at any given time. The ideas behind bitcoin have been further generalized: blockchains enabling smart contracts such as Ethereum constitute even more powerful types of distributed database.

But bitcoin and other proof of work-based systems have one major drawback: energy consumption. One source [1] reports that the power consumed for bitcoin proof-of-work in January 2019 was around 40TWh/year, comparable to the power use of a small country. In essence, the public trust that bitcoin guarantees is directly correlated to the energy consumption of the bitcoin miners. To put it differently, bitcoin's resiliency to attack is a direct result of consuming large amounts of power.

This brings us to a fundamental question in modern distributed databases: can we build systems with many of the good core properties of bitcoin–scalability and broadly decentralized trust—without the drawbacks associated with mining?

In an attempt to offer a low power but scalable alternative, Intel included in the Hyperledger Sawtooth [2] distributed ledger platform a form of Nakamoto consensus that replaced proof-of-work with an alternative called *proof of elapsed time* (*PoET*) [3], which utilizes the security properties provided by a trusted execution environment (TEE). Several academic works [21,47] point to the efficiency of PoET and its strong performance in large systems. However, PoET lacks any formal analysis, and we fill that gap in this paper.

1.1 Our Contributions

We generalize the PoET and Proof of Luck protocols into what we call *elapsed time (ET) consensus*, where we relax the PoET protocol to focus on the critical protocol itself and ignore some of the difficulties faced in practical implementations, such as bootstrapping, onboarding parties, and dynamic membership. We provide a formal description of our ET consensus protocol (as per our knowledge, there is no formal description of PoET available anywhere).

Our model can be considered as a generalization of the bitcoin backbone model [28,29]—that might be of independent interest.

We focus on elapsed time consensus protocols with two main assumptions on the TEE: (1) The TEE has access to a trusted timer. (2) No TEEs are present, or TEEs can be easily compromised [32,34,36,49].

ET Consensus with Trusted Timer. We first define a basic version of our protocol: *elapsed time consensus with a trusted timer*, where the TEE has access to a trusted timer.[1] This protocol captures the essence of PoET [3] and other related works like proof of luck [40] (which we discuss more later). We show that our ET consensus protocol with a trusted timer provides similar security guarantees as bitcoin with the same honest majority assumption.

Elapsed Time Consensus with a z-Test. We modify our basic ET consensus protocol and include a z-test to avoid the dependency on TEEs. However, our z-test is quite different than the one proposed by the Intel engineers, as theirs is not sufficient for our proof.[2] Instead of checking the proportion of the total number of blocks a player produces in a chain, we (essentially) restrict how many blocks a player can produce over a sliding window of time.

We prove that ET consensus with our z-test is secure in our (permissioned) model **without TEEs** and assuming that some constant fraction ($> \frac{2}{3}$) of the participants are Byzantine. In other words, we show that "bitcoin without mining" coupled with some clever rate-limiting (the z-test) is secure in a permissioned network without any hardware security guarantees, even if up to 33% parties are dishonest.

Until now, PoET, proof of luck, and other elapsed time based consensus systems have never (to our knowledge) been formally proven secure. Perhaps the most exciting implication of our proofs, though, is that we can ignore TEEs completely in these protocols and still maintain relatively good security with our z-test. This notion of "permissioned bitcoin without mining" will be very useful for future blockchain developments.

1.2 Related Work

The only current work of which we are aware on the security of PoET is [19]. This work shows how an adversary that is capable of compromising SGX can attack PoET up to the bounds of the z-test that PoET currently uses. Unfortunately, this paper does not offer any formal analysis in the other direction: it does not include a rigorous security proof that PoET is secure outside of these bounds. As

[1] Many TEEs, including SGX [4], have a trusted timer or equivalent functionality.

[2] Using the z-test given by the PoET specification would require much stronger parameter settings than for which we achieve provable security.

we have mentioned before, we note that Milutinovic et al. [40] define a consensus protocol called *proof of luck* (PoL) that functions very similarly to PoET, but do not offer a security analysis to their protocol or even a comparison to PoET. Additionally, the authors of [5] show a construction of a proof of stake protocol using TEEs, but this protocol also lacks a formal security proof.

Improved consensus algorithms and models with exciting new properties have proliferated recently. Some examples include Thunderella [45], the sleepy model of consensus [44], Snow White [22], Fruitchain [43], Ouroboros [24,31], Bitcoin-NG [27], Casper [15], Stellar [39], and ByzCoin [33]. Exciting new work in the space includes things like proofs of space and storage [9,20,26] and verifiable delay functions [12]. However, most of these protocols focus on public blockchains.

Comparatively, the academically-focused work on permissioned blockchains has been substantially less, but has notably included things like an analysis of Hyperledger Fabric [6,16] and the Tendermint consensus protocol [35]. Work on Byzantine fault tolerant consensus has also been done [11], including the recent an exciting development of [51]. There have also been a number of very useful papers that have analyzed these consensus protocols and their properties, including [10,17,30,42,50].

Several consensus protocols in past have leveraged different forms of trusted hardware. For example, MinBFT [46] proposes a trusted counter to reduce the number of nodes required from $3F + 1$ to $2F + 1$ for F faulty nodes. More recently, FastBFT [37] uses a trusted execution environment to aggregate messages for latency and throughput improvements. Other protocols use TEEs for the purposes of sharing on blockchains [23].

2 Elapsed Time Backbone Model

Our model can be considered as a generalization of [28]. Here we provide a summary and refer to our extended version [13] for a complete description.

2.1 Notations

We assume that the blockchain protocol has a fixed number n of players. We use $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ to represent a cryptographic hash function (modelled as a random oracle). A *block* is any tuple of the form $B = \langle s, x, \pi \rangle$, where $s \in \{0,1\}^\kappa$ is the hash of the previous block, $x \in \{0,1\}^*$ is the `content` of the block and $\pi \in \{0,1\}^*$ is the proof of block validity. `validblock` is the predicate that takes a block B and a chain $\mathcal{C}$ as input, checks validity of the content of B.

A *blockchain* or *chain* is a sequence of *blocks*. The (current) last block of a blockchain $\mathcal{C}$ is called the *head* of $\mathcal{C}$ and is denoted by $\mathcal{C}.head$. For an empty string or empty blockchain ε, we have $\varepsilon.head = \varepsilon$. The *length* of a chain is its number of blocks. A chain $\mathcal{C}$ can be extended by a block $B = \langle s, x, \pi \rangle$ if $s = \mathcal{H}(\mathcal{C}.head)$, $\texttt{ValidBlock}(B, \mathcal{C})$ is `true` and $B.\pi$ is a valid proof for the extended chain $\mathcal{C}' = \mathcal{C} \| B$. For the new chain $\mathcal{C}'$, we have $\mathcal{C}'.head = B$. For any chain $\mathcal{C} = (B_1, \cdots, B_\ell)$, $Length(\mathcal{C}) = \ell$ denotes the length of the chain. For any pair of integers $1 \leq i \leq j \leq \ell$, the chain $\tilde{\mathcal{C}} = (B_i, \cdots, B_j)$ is called a subchain of $\mathcal{C}$. $\mathcal{C}^{\lceil k}$ denotes the chain resulting from removing the k rightmost blocks, for a given non-negative integer k. If $k \geq \ell$, then $\mathcal{C}^{\lceil k} = \varepsilon$. For two chains $\mathcal{C}_1, \mathcal{C}_2$ the notation $\mathcal{C}_1 \preceq \mathcal{C}_2$ denotes that $\mathcal{C}_1$ is a prefix of $\mathcal{C}_2$.

2.2 Model and Structure

Our model assumes round-based protocols as in [7,28,29]. In our model we have three top level parameters: the total number of players n, the security parameter λ and honest parties' advantage δ. If t is the number of corrupt players then we require that $\frac{t}{n-t} \leq 1 - \delta$.

Adversary. We allow the adversary to see all of the *global variables* in the system, including the description of all of the algorithms in the protocol. The adversary can corrupt players at the beginning or during the protocol run, as long as the total number of corrupted players does not exceed t.

Model Rules. We model communication among protocol parties by having a global array of queues $PLAYER\_QUEUE[n]$ - where $PLAYER\_QUEUE[i]$ represents the queue associated with player i. We let each party send to other players' queues, but only read from their own queue.

2.3 Abstractions

In this work we use the following abstractions to simplify our protocol and proof.

Certifications. We use the *Cert* functionality (Fig. 1) instead of explicitly using digital signatures - they act as completely unforgeable, perfect digital signatures.

```
$\mathbf{Cert_u}$ (Statement $m$): //certificate issued by $u$ attesting validity of $m$

$\mathbf{IsCertValid}_u$ ($Cert\ \ c_{u,m}$, **Statement** $m$):
    Return 1 if $c_{u,m}$ certifies statement $m$ on behalf of user $u$ and 0 otherwise.
```

Fig. 1. Certification functionality.

Trusted Execution Environment (TEE). In our model, a TEE is an unbreakable black box (a VBB obfuscator [8]) that runs some code in a way that completely hides the internals of the code—an adversary can only see the input and output values of the program being run by the TEE. We note that this is an idealistic model for a TEE since it presumes perfect security and no side channel attacks. We represent our abstraction of TEE in Fig. 2.

In some protocols, we will give our TEE a trusted timer functionality. Since time is approximated by round number in our protocol, we will have the TEE return the current round number for this function. Additionally, we endow our TEE with a monotonic counter *TEE.Counter* which can never be decreased in value, *even if the TEE is reset.* We note that many modern TEEs like Intel's SGX have both trusted timer and monotonic counter functionalities [4].

```
**TEE.**$\alpha$ := $null$;  // Can be set to any program code.
**TEE.**$args$ := $null$;  // Arguments for $\alpha()$.
**TEE.**$Counter$ := 0;  // monotonic counter inside TEE.

**TEE.GetCounterValue**()
  **Return** TEE.Counter    // Just return the counter value.

**TEE.CounterSet**($x$)
  **if** $TEE.Counter < x$ **then** $TEE.Counter \leftarrow x$ **end if**

**TEE.Run**($Prog, arguments$):
  // Run the program $Prog$ inside the TEE.
  // If there is any currently running program it aborts the current program.

  Abort($\alpha$); $\alpha \leftarrow Prog$; $args \leftarrow arguments$; Run $\alpha(args)$

**TEE.Poll():**
  // Can be called only after calling TEE.Run()
  **if** $\alpha = null$ **then** return $\perp$ **end if**
  **if** $\alpha()$ has completed running **then**
    $O \leftarrow$ output of $\alpha(args)$, **Return** $(O, \mathbf{Cert}_{TEE}(O||\alpha||args))$
  **else** return $Incomplete$ **end if**
```

Fig. 2. TEE functionality

2.4 Blockchain Properties

We next define three core blockchain properties: the common prefix property, the chain quality property, and the chain growth property. As argued in [28], these properties together essentially define what it means to be a functional and useful blockchain.

Definition 1. ***Common Prefix Property****: Suppose $\mathcal{C}_1$ is a chain which has been accepted by an honest party at round r_1 and $\mathcal{C}_2$ is another chain which has been accepted by some honest party at round $r_2 (\geq r_1)$. Then $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ holds for all integers $k \geq \ell_{cf}$, where ℓ_{cf} is the common prefix parameter.*

Definition 2. ***Chain Quality Property****: Suppose $\mathcal{C}$ is a chain that has been accepted by some honest party. Any subchain $\tilde{\mathcal{C}}$ of $\mathcal{C}$ of length $\tilde{\ell} \geq \ell_q$ must contain at least $\mu\tilde{\ell}$ many honest blocks. Here ℓ_q, μ are the chain quality parameters (Table 1).*

Table 1. Table of all parameters

n	:	Total number of players
t	:	Number of corrupted players
δ	:	Advantage of honest parties, $(\frac{t}{n-t} \leq 1-\delta)$
p	:	Probability that an honest player creates a block in a given round
f	:	Probability at least one honest player creates a block in a given round
r_{end}	:	Total number of rounds in the security game
ϵ	:	A security parameter used to bound the "luckiness" of the adversary See the "typical execution" definition in Sect. 5.1
ϵ'	:	Quality of concentration for z-test (Definition 4)
λ	:	Security parameter
ℓ_{cf}	:	Minimum number of blocks in common prefix property
μ	:	Parameter in chain quality property
ℓ_q	:	Minimum number of blocks for which the chain quality property holds.
τ	:	Minimum number of blocks in chain growth property.
σ	:	Maximum number of blocks in chain growth property.
r_g	:	Minimum number of rounds for which the chain growth property holds

A block is an honest block, if it is created by an honest party.

Definition 3. ***Chain Growth Property:*** *Suppose $\mathcal{C}_1$ is a chain of length ℓ_1 which has been accepted by an honest party at round r_1 and $\mathcal{C}_2$ is another chain of length ℓ_2 which has been accepted by some honest party at round r_2. If $r_2 - r_1 > r_g$, then $\ell_2 - \ell_1 \in [\tau(r_2 - r_1), \sigma(r_2 - r_1)]$. Here r_g, τ, σ are the chain growth parameters.*

Block.π:

π.timestamp // Round of "mining."
π.WaitTime // How many rounds until the block can be issued.
π.WaitCert // Proof that a TEE generated the WaitTime properly.

Fig. 3. Block proof structure

3 Elapsed Time Consensus Protocol with Trusted Timer

Here we construct an elapsed time consensus protocol with a trusted timer. We start by defining the block validity proof π of a block in Fig. 3. The basic version of our ET protocol (with trusted timer), denoted by ET_{timer}, works in two phases: (1) Initialization phase and (2) Leader election phase.

Initialization Phase. In this phase, the challenger initializes the blockchain by calling $Genesis()$ (refer to Fig. 4), which in turn creates the genesis block $B_{genesis}$ and initializes all the players.

```
**Genesis**$(n, \lambda)$:
    generate the genesis block $B_{genesis}$
    $P$ := Geometric Distribution with parameter $p$
    **for** $i = 1$ to $n$; i++ **do**
        $PLAYERS[i].Initialize(i, B_{genesis})$
    **end for**
```

Fig. 4. Initialization of the *Blockchain* for ET_{timer}

Leader Election Phase. In this phase, all the players compete with each other to be elected to generate the next block. Once a block is generated, they start a fresh competition and repeat the process [3]. We define the leader election protocol in the $RunPlayer()$ routine defined in Fig. 5. For an honest user, the challenger runs $RunPlayer()$ exactly once per round.

In every round, $RunPlayer()$ checks if there are new chains in the input buffer of the player. These chains are generated in the previous round by other players. Our current player picks the best chain according to $PickChain()$

```
chain $\mathcal{C}$ := empty; // A chain - an ordered list of blocks.
$playerID$; // denotes the index of the player provided by the challenger
$TEE$; // denotes the trusted execution environment specific to the player

**Initialize( integer** $i$**, the genesis block** $B_{genesis}$**):**
    $playerID \leftarrow i$ ; Add $B_{genesis}$ to $\mathcal{C}$;
    Initialize the $TEE$ for player $i$
    $x \leftarrow$ Collected transactions from users;
    $TEE.Run(WaitForBlock(), \mathcal{C}, x)$

**RunPlayer():**
    **if** $PLAYER\_QUEUE[playerID]$ is not empty **then**
        $\mathcal{C}_{new} \leftarrow PickChain(playerID, \mathcal{C})$ // Defined in Fig 6
        **if** $\mathcal{C}_{new} \neq \mathcal{C}$ **then**
            $\mathcal{C} \leftarrow \mathcal{C}_{new}$; $x \leftarrow$ Collected transactions from users
            $TEE.Run(WaitForBlock, \mathcal{C}, x)$    // Wait on TEE for the next block
        **end if**
    **else**
        **if** $TEE.Poll() \neq Incomplete \wedge TEE.Poll() \neq \perp$ **then**
            // It seems the player is a leader
            $(B, WaitCert) \leftarrow TEE.Poll()$; $B.\pi.WaitCert \leftarrow WaitCert$
            Add $B$ to $\mathcal{C}$; $Broadcast(\mathcal{C})$
            // Again, repeat for the next leader election
            $x \leftarrow$ Collected transactions; $TEE.Run(WaitForBlock, \mathcal{C}, x)$
        **end if**
    **end if**
```

Fig. 5. Ideal functionality of ET consensus protocol

(defined in Fig. 6) to replace (if applicable) its own local chain. If $PickChain()$ updates the local chain, our player stops competing for the last election and calls $TEE.Run(WaitForBlock, \mathcal{C}, \cdots)$ to start a new competition.

```
PickChain(integer $i$ denoting the player index, current chain $\mathcal{C}$):
  for each $\mathcal{C}^*$ in $PLAYER\_QUEUE[i]$ do
    // Check if the chain is valid and longer than the current chain
    // In case of a tie, keep the current chain
    if $(Length(\mathcal{C}^*) > Length(\mathcal{C})) \wedge IsChainValid(\mathcal{C}^*)$ then $\mathcal{C} \leftarrow \mathcal{C}^*$ end if
  end for
  return $\mathcal{C}$

IsChainValid( chain $\mathcal{C}^* = \{B_1, B_2, \ldots B_\ell\}$ ):
  if $B_1$ is not the genesis block then return $false$ end if
  for each block $B_i$ in $\mathcal{C}^*$ except $B_1$ do
    $\mathcal{C}_{i-1} \leftarrow (B_1, \cdots, B_{i-1})$
    // Verify that the current block is pointing to the previous block
    if ( $B_i.s \neq \mathcal{H}(B_{i-1})$ ) then return $false$
    // Verify that the block is not created ahead of its previous block
    else if $(B_i.\pi.timestamp < B_i.\pi.WaitTime + B_{i-1}.\pi.timestamp)$
    then return $false$
    // Verify the $WaitCert$ and content for the block
    else if $!ValidBlock(B_i, \mathcal{C}_{i-1}) \vee$ !IsCertValid$_{TEE}(B_i.\pi.WaitCert,$
                                  $B_i.\pi.WaitTime \| WaitForBlock \| (\mathcal{C}_{i-1}, B_i.x))$
    then return $false$
    else continue
    end if
  end for
  // Verify that the last block is not created ahead of time
  if ( $B_\ell.\pi.timestamp \geq$ current-round ) then return $false$ end if
  // If z-test is running, verify that the chain satisfies z-test property
  if ( $z_{\epsilon',\lambda}(\mathcal{C}, B_\ell.\pi.playerID) = 0$ ) then return $false$ end if
  return $true$
```

Fig. 6. Chain selection mechanism for ET consensus (the part in red is only applicable when z-test is used as defined in Definition 4.)

Given an existing chain $\mathcal{C}$, the leader election mechanism works very much like a lottery algorithm: a player wins an election if their $WaitTime$ (picked from a pre-defined probability distribution P) is smallest among all players. The TEE runs $WaitForBlock(\mathcal{C})$ (defined in Fig. 7) for a player to generate $WaitTime$.

Collision Resolution: It is possible that two players get the same $WaitTime$. In that case, both the players broadcast their chains in the same round, each chain with a newly added block – it is considered a collision.

Each player handles collision locally in the following way (refer to Fig. 6 for the pseudocode representation): If two chains $C1$ and $C2$ are of same length, and the last blocks in $C1$ and $C2$ were generated at rounds t_1 and t_2 respectively. Among t_1 and t_2, whichever is smaller the corresponding chain is chosen. However, if $t_1 = t_2$, the collision is not resolved; the player can choose any one of the chains as the current chain and keeps mining for that chain. If C_1 and C_2 are of different lengths, the longer chain is chosen by the player.

Note that, since we have a synchronized round based model and the players have access to a reliable broadcast mechanism, it might seem unlikely to have $t_1 \neq t_2$. However, a compromised player might choose not to broadcast his chain, or selectively send the chain to some players.

```
**WaitForBlock(chain $\mathcal{C}$, transactions $x$)**
  if (TEE.GetCounterValue() $\geq$ $Length(\mathcal{C})$) then return $\perp$
  else TEE.CounterSet($Length(\mathcal{C})$) end if
  $waitTime \leftarrow$ draw an element from $P$; $sleep(waitTime)$
  $B \leftarrow CreateBlock(waitTime, \mathcal{C}, x)$; return $B$

**CreateBlock(time delay $t$, chain $\mathcal{C}$, transactions $x$):**
  create an empty block $B$; $B.x \leftarrow x$    // Add transactions
  $B.s \leftarrow \mathcal{H}(\mathcal{C}.head)$    // Point to the last block of the existing chain
  $B.\pi.timestamp \leftarrow$ current-round; $B.\pi.WaitTime \leftarrow t$
  $B.\pi.playerID \leftarrow$ current player ID; return $B$
```

Fig. 7. Function to be run under Trusted Execution Environment (TEE)

Probability Distribution for WaitTime. In our protocol, the TEE generates the $WaitTime$ by sampling from a probability distribution. In our case, we use a geometric distribution where the probability of success in each round is p.[3] The parameters of the probability distribution are defined by the protocol, and are globally known. Each $WaitTime$ is independent of all previous ones and all other players. We denote the probability distribution with P.

Due to the memorylessness of geometric distribution adversarial parties gain no information about the sampled wait times until the wait time actually expires and TEE releases the block. The following lemma captures this fact.

Lemma 1. *The event that any party generates a block in a given round happens with probability at most p over the random coins of that party's TEE. This event is independent of all other random coins in the protocol as well as adversarial choices.*

Proof (Proof Sketch for Lemma 1).

The probability of block generation by an honest party can only be influenced if a new block arrives. However, because the adversary has no knowledge about

[3] Here we are considering the version of geometric distribution where trial number $WaitTime$ is the first successful trial.

the current wait times of honest or adversarial parties, drawing a new wait time is exactly same as drawing a new element from the probability distribution with replacement, and hence, is independent of the current wait times or any past wait times.

The memorylessness property of geometric distribution ensures that the probability of generating a block by an honest party is p in a given round, independent of if the party has queried a new wait time in that round or not. An adversarial party can draw a new wait time in two ways:

1. if the party decides to discard the current wait time and draw a new wait time,
2. or, after the completion of the current wait time.

In the first case, the probability for the adversarial party to generate a block in a given round still remains p since the wait time is stored inside the TEE and the party does not see it. In the second case, the probability is trivially p. Note here, we assume that the adversarial party always generates a block, when the TEE time expires. □

With the above lemma in place the security proof for ET consensus with trusted timer is very similar to that of bitcoin [29] and provides exactly the same security properties; we skip the detailed proof here and refer to [29]; in Appendix 6.4 we summarize the security properties achieved by the protocol for completeness.

4 ET Consensus with Z-Test

We already know that operations within a TEE may be subject to attacks [32, 34, 36, 49]. In PoET, PoL, and ET consensus with TEEs, we are mainly concerned about two forms of attack: accelerating the trusted timer and impersonating (or stealing the secret keys from) a TEE. In both cases, an attacker can create an invalid leadership claim and break the security of the protocol.

To address this issue, PoET implements a "z-test" (or more accurately, it implements a "1 sample z-test") that limits the number of blocks any validator can win in some (large) consecutive set of blocks. The "z-test" is based on the observation that while any validator can win any block (the fairness principle), the probability that a particular validator wins a disproportionately large number of blocks is extremely low.

4.1 Our ET Consensus Protocol with z-Test

While the PoET z-test is an excellent innovation, we cannot prove the security of any ET consensus protocols using that z-test. We implement a slightly different z-test—PoET rate-limits what percentage of blocks any given participant can win on the entire chain, while we rate-limit each participant's block wins over a sliding window of time (or, in our protocol, a sliding window of rounds). An adversary can decide not create blocks for a long time, then create enough in a short window to violate consensus while bypassing the z-test by PoET; however, our z-test would catch such attacks. Given the distribution of block creation, we can figure out the expectation of (and appropriate other statistics around) how many blocks a player wins over a particular time period. Deviating too far from this results in future blocks being declared invalid.[4]

For a consecutive set of rounds S, we denote number of adversarial blocks in a chain $\mathcal{C}$ by $ADV_C(S)$. In other words, $ADV_C(S) = |\{B \in \mathcal{C} : \texttt{round}(B) \in S \text{ and } \texttt{id}(B) \text{ is corrupted. }\}|$. For a player $i \in [1, n]$, $Z_{\mathcal{C}}(i, S)$ denotes the number of blocks in chain $\mathcal{C}$ produced by player i, created in rounds in S, i.e. $Z_{\mathcal{C}}(i, S) = |\{B \in \mathcal{C} : \texttt{round}(B) \in S \text{ and } \texttt{id}(B) = i\}|$. Hence, for a chain $\mathcal{C}$ and set of rounds S, we have $ADV_{\mathcal{C}}(S) = \sum_{\substack{i \in [1,n] \\ i \text{ is corrupted}}} Z_{\mathcal{C}}(i, S)$.

Definition 4. *Let $\mathcal{C}$ be a chain. For $\epsilon' \in (0, 1)$ and $\lambda > 0$ the function $z_{\epsilon',\lambda} : \mathcal{C} \times [1, ..., n] \to \{0, 1\}$ be defined in the following way:*

$$z_{\epsilon',\lambda}(\mathcal{C}, i) = \left\{ \begin{matrix} 0 & \text{if } \exists \text{ set of consecutive rounds } S \text{ s.t.} \\ & |S| \geq \lambda \text{ and } Z_{\mathcal{C}}(i, S) > (1 + \epsilon')p|S| \\ 1 & \text{otherwise} \end{matrix} \right\} \tag{1}$$

$z_{\epsilon',\lambda}(\mathcal{C}, i) = 0$ if a party i has contributed more than the allowed number of blocks in chain $\mathcal{C}$.

We set our z-test parameters so that honest parties will only be affected with negligible probability, so our z-test has no negative impact on honest blockchain operation. Our new z-test can effectively stop an adversary from concentrating a high number of blocks in a very small amount of time and allows us to state concrete facts about the security of our protocol with the z-test.

4.2 Modification in the Protocol with Z-Test

In this section, we do not assume any integrity of the TEE. Hence, adversarial parties can generate arbitrarily many valid blocks per round. However, honest parties apply the 'z-test' before accepting a chain, which checks that no single player is producing substantially more than their fair share of the blocks.

Although we do not assume any security of any TEE (or even the existence of a TEE), the probability any honest party generates a block in any given round still remains p. This holds because adversarial parties cannot influence the wait time distributions of honest parties. We capture this in the following lemma. We skip the proof here because it is very similar to the proof of Lemma 1.

Lemma 2. *The event that any honest party generates a block in a given round happens with probability p, over the random coins of that party's TEE. This event is independent of all other random coins in the protocol as well as adversarial choices.*

[4] We also note that this change helps us to address the attacks in [19].

5 Security of ET Consensus with Z-Test

Even though z-test is a powerful assumption, the adversary can still essentially allocate their blocks however they want over the given time periods. In addition, the adversary can also create many different small forks or chains like a "nothing at stake" attacker [14]. It is important to note that the z-test bounds the behaviour of the adversary on *each* chain, not globally. Therefore, we use a slightly modified honest majority assumption (where we require slightly more honest parties) compared to bitcoin, and prove that honest parties are "stronger" than the adversary on each *valid* chain.

To prove the desired security properties for ET_{ztest}, we assume cryptographic security of the hash function and the signature scheme, and an honest majority assumption (formally stated below).

Definition 5 (Honest Majority Assumption). *Suppose* n *is the total number of parties, and out of them* t *parties are corrupted. Then we require that* $t < (1-\delta)(n-t)$, *where* $\max\left(\frac{2f+\epsilon+\epsilon'-f^2-f\epsilon}{1+\epsilon'}, \frac{1+\epsilon+2\epsilon'-2\epsilon' f-2\epsilon f}{2(1+\epsilon')(1-f)}\right) < \delta \leq 1$, $\epsilon \in (0,1)$ *and* $\epsilon' > 0$.

We shall use the following boolean random variables for the proofs:

- $HON_i^{\geq 1}$ is defined to be 1 if *at least* one honest party creates a block at round i and 0 otherwise.
- HON_i^1 is defined to be 1 if exactly one *single* honest party creates a block at round i and 0 otherwise.
- $ADV_{i,j}$ is defined to be 1 if the jth dishonest party creates a block at round i and 0 otherwise[5]. We also define, $ADV_i := \sum_j ADV_{i,j}$.
- $HON_{i,j}$ is defined to be 1 if jth honest player successfully generated a block at round i and 0 otherwise. We denote $HON_{\cdot,i}(S) = \sum_{r\in S} HON_{r,i}$.

Below, we mention an inequality that we will use often.

$$f < p(n-t) < \frac{f}{1-f} \tag{2}$$

Note, the first inequality is a straight forward application of Bernoulli's inequality which says for real $x > -1$ and integer $r \geq 0$ we have, $(1+x)^r > 1+rx$. The second inequality is another application of Bernoulli's inequality after applying the following inequality $(1-p)^{-(n-t)} > (1+p)^{n-t}$.

[5] The adversary actually has a choice to try to mine in a specific round or not. However, without loss of generality we can consider an adversary who always tries to do so, as the adversary is always free to discard a successfully mined block. This assumption helps us in defining random variables $ADV_{i,j}$ in terms of pure probabilistic events.

5.1 Typical Execution

We slightly modify the definition of typical execution from the bitcoin backbone work [28] and use here. More specifically, we do not impose any condition on adversarial success, because without integrity of the trusted execution environment the adversary is free to generate as many valid blocks as it wants (although, if they generate too many, they will be rejected by honest parties using the z-test property).

Definition 6 (Typical Execution). *An execution of r_{end} rounds, is $(\epsilon, \epsilon', \lambda)$-typical, for some $\epsilon \in (0,1), \epsilon' > 0$, if for any set S of at least λ consecutive rounds the following hold:*

(a) $(1-\epsilon)\mathbb{E}[HON^{\geq 1}(S)] < HON^{\geq 1}(S) < (1+\epsilon)\mathbb{E}[HON^{\geq 1}(S)]$ *and* $(1-\epsilon)\mathbb{E}[HON^{1}(S)] < HON^{1}(S)$
(b) For all honest players i, $HON_{\cdot,i}(S) < (1+\epsilon')\mathbb{E}[HON_{\cdot,i}(S)]$

Theorem 1. *An execution of r_{end} rounds is $(\epsilon, \epsilon', \lambda)$-typical with probability at least $1 - r_{end}(e^{-\Omega(\epsilon^2 \lambda f)} + (n-t)e^{-\Omega(\epsilon'^2 \lambda p)})$.*

Proof (Sketch). Note, there are $(n-t)$ honest parties and for every honest party i, we have $\mathbb{E}[HON_{\cdot,i}(S)] = p|S|$. The theorem follows with a Chernoff bound. □

Now, let us look at some more properties of a typical execution under z-test. Later in this section, we are going to use those properties to analyze the chain growth and chain quality properties.

Lemma 3. *For any set S of at least λ consecutive rounds, the following properties hold in a typical execution where $\mathcal{C}$ is a chain adopted by an honest party.*

(a) $(1-\epsilon)f|S| < HON^{\geq 1}(S) < (1+\epsilon)f|S|$
(b) $ADV_{\mathcal{C}}(S) < \frac{(1+\epsilon')t}{(n-t)(1-f)(1-\epsilon)} HON^{\geq 1}(S)$
(c) $2ADV_{\mathcal{C}}(S) < HON^{1}(S)$
(d) $ADV_{\mathcal{C}}(S) < (1-f-\epsilon)f|S|$

Proof. Recall, $\mathbb{E}[HON^{\geq 1}(S)] = f|S|$. Hence, part (a) readily follows from definition of typical execution (Definition 6). The chain $\mathcal{C}$ got adopted by an honest party, hence it passed the 'z-test' for all parties. As $|S| \geq \lambda$, for all players $i \in [1,n]$ we have $Z_{\mathcal{C}}(i,S) < (1+\epsilon')p|S|$. Hence,

$$ADV_{\mathcal{C}}(S) = \sum_{\substack{i \in [1,n] \\ i \text{ is corrupted}}} Z_{\mathcal{C}}(i,S) < (1+\epsilon')pt|S| < \frac{(1+\epsilon')tf}{(n-t)(1-f)}|S| \qquad (3)$$

The last inequality above uses inequality (2). Now we can prove part (b) by applying $HON^{\geq 1}(S)$ lower bound from part (a). For part (c), from the definition of typical execution we have,

$$\begin{aligned} HON^{1}(S) &> (1-\epsilon)\mathbb{E}[HON^{1}(S)] \\ &= (1-\epsilon)(n-t)p(1-p)^{n-t-1}|S| \\ &> (1-\epsilon)(n-t)p(1-p)^{n-t}|S| \end{aligned}$$

$$\begin{aligned}
&> (1-\epsilon)(n-t)p(1-p(n-t))|S| && \text{By Bernoulli's inequality}\\
&> (1-\epsilon)(n-t)p(1-\frac{f}{1-f})|S| && \text{By Inequality 2}\\
&= \frac{(1-\epsilon)(1-2f)}{(1+\epsilon')(1-f)}\frac{(n-t)}{t}(1+\epsilon')pt|S| &&\\
&> \frac{(1-\epsilon)(1-2f)}{(1+\epsilon')(1-f)}\frac{(n-t)}{t}ADV_{\mathcal{C}}(S) && \text{By Inequality 3}\\
&> \frac{(1-\epsilon)(1-2f)}{(1+\epsilon')(1-f)(1-\delta)}ADV_{\mathcal{C}}(S) && \text{Definition 5}
\end{aligned}$$

From the honest majority assumption or Definition 5 we also have

$$\frac{1+\epsilon+2\epsilon'-2\epsilon' f-2\epsilon f}{2(1+\epsilon')(1-f)} < \delta < 1,$$

which in turn implies $\frac{(1-\epsilon)(1-2f)}{(1+\epsilon')(1-f)(1-\delta)} > 2$. This completes the proof of part (c). For part (d), from Inequality (3) we have

$$\begin{aligned}
ADV_{\mathcal{C}}(S) &< (1+\epsilon')pt|S| &&\\
&< (1+\epsilon')(1-\delta)p(n-t)|S| && \text{By Definition 5}\\
&< (1+\epsilon')(1-\delta)\frac{f}{1-f}|S| && \text{By Inequality (2)}\\
&= \frac{(1+\epsilon')(1-\delta)}{(1-f)(1-f-\epsilon)}(1-f-\epsilon)f|S| &&
\end{aligned}$$

From honest majority assumption we also have $\frac{2f+\epsilon+\epsilon'-f^2-f\epsilon}{1+\epsilon'} < \delta < 1$, which in turn implies $\frac{(1+\epsilon')(1-\delta)}{(1-f)(1-f-\epsilon)} < 1$. This completes the proof of part (d). □

5.2 Chain Growth Properties

Now, we want to prove the chain growth property for ET consensus with z-test. However, the property crucially depends upon the fact that in a typical execution, honestly generated blocks never get rejected because of the 'z-test'. We note that this property is easy to achieve with what should be fairly typical parameter settings: namely, with $\epsilon \leq \epsilon'$.

Lemma 4. *In a typical execution, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two chains which were adopted by some honest parties. Suppose B_1 and B_2 are the k-th blocks of chains $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. If $\mathtt{id}(B_1)$ is an honest user and $\mathtt{round}(B_1)$ is a uniquely successful round, then either $B_1 = B_2$ or $\mathtt{id}(B_2)$ is corrupted.*

Proof. For contradiction, we assume $B_1 \neq B_2$ and $\mathtt{id}(B_2)$ is honest. Security of the signature scheme implies the blocks B_1 and B_2 are actually created by $\mathtt{id}(B_1)$ and $\mathtt{id}(B_2)$. As, $\mathtt{round}(B_1)$ is uniquely successful round and both $\mathtt{id}(B_1)$

and $\texttt{id}(B_2)$ are honest, we have $\texttt{round}(B_1) \neq \texttt{round}(B_2)$. Suppose, $\texttt{round}(B_1) < \texttt{round}(B_2)$; as both of the players are honest $\texttt{id}(B_2)$ must have received the chain ending in B_1 with length k on or before $\texttt{round}(B_2)$. This implies position of the block B_2 must be greater than k, which is a contradiction. The cryptographic security of the hash function ensures an honest party creates a block at position k and that adversarial players cannot insert that block at a different position. A similar argument holds for the case $\texttt{round}(B_1) < \texttt{round}(B_2)$. □

Lemma 5 (Chain Growth Lemma). *In a typical execution, suppose an honest party has adopted a chain of length ℓ at round r. Then, by round $h > r$, every honest party has adopted a chain of length at least $\ell + \sum_{i=r}^{h-1} HON_i^{\geq 1}$.*

Proof. We prove the above theorem using induction on $h \geq r+1$.

Induction base: The protocol has moved only one round after round r, hence $h = r+1$. If at round r, an honest party has a chain of length ℓ, every honest party will adopt a chain of length at least ℓ by round $r+1$. Additionally, if $HON_r^{\geq 1} = 0$, the statement follows directly. If $HON_r^{\geq 1} = 1$, the successful honest party will broadcast a chain of length $\ell + 1 = \ell + HON_r^{\geq 1}$, and all honest parties will adopt a chain of at least that length by round $h = r+1$.

Inductive step: Let us assume that every honest party has adopted a chain of length at least $\ell' = \ell + \sum_{i=r}^{h-2} HON_i^{\geq 1}$ by round $h-1$.
Now, two things could have happened on round $h-1$:

1. $HON_{h-1}^{\geq 1} = 0$, in which case $\sum_{i=r}^{h-1} HON_i^{\geq 1} = \sum_{i=r}^{h-2} HON_i^{\geq 1}$. Hence, the statement follows.
2. $HON_{h-1}^{\geq 1} = 1$, in that case a successful honest party will broadcast a chain of length at least $\ell' + 1$ in round $h-1$. By round s, all honest parties will adopt a chain of length at least $\ell' + 1 = \ell + \sum_{i=r}^{h-2} HON_i^{\geq 1} + 1 = \ell + \sum_{i=r}^{h-1} HON_i^{\geq 1}$.

□

Lemma 6 (Chain Growth Upper Bound). *Suppose $\mathcal{C}$ is a chain adopted by an honest party during a typical execution. For any $k \geq \max(2\lambda f, 4)$, let $B_m, B_{m+1}, \cdots, B_{m+k-1}$ be k consecutive blocks of the chain $\mathcal{C}$. Then, we have*

$$|[\textit{round}(B_m), \textit{round}(B_{m+k-1})]| \geq \frac{k}{2f}.$$

Proof. Suppose $S' = [\texttt{round}(B_m), \texttt{round}(B_{m+k-1})]$. For contradiction let us assume $|S'| < \frac{k}{2f}$. Consider the set S of consecutive rounds such that $S \supseteq S'$ and $|S| = \lceil \frac{k}{2f} \rceil$. Security of the signature scheme along with the fact chain $\mathcal{C}$ has been adopted by an honest party ensures $HON^{\geq 1}(S') + ADV_{\mathcal{C}}(S') \geq k$. As

$S' \subseteq S$, this in turn implies $HON^{\geq 1}(S) + ADV_{\mathcal{C}}(S) \geq k$. As, $|S| \geq \lambda$, we can apply Lemma 3 and it implies the following.

$$\begin{aligned}
&HON^{\geq 1}(S) + ADV_{\mathcal{C}}(S) \\
&< (1+\epsilon)f|S| + (1-f-\epsilon)f|S| \\
&< (2-f)f|S| \\
&< (2-f)f(\frac{k}{2f}+1) && \text{Since } |S| = \lceil \frac{k}{2f} \rceil < \frac{k}{2f}+1 \\
&\leq k - \frac{kf}{2} + 2f - f^2 < k + f(1-k/4) < k. && \text{Since } k \geq 4
\end{aligned}$$

This shows we have a contradiction. □

Lemma 6 provides us an upper limit on the rate of chain growth. It says that at least $\frac{k}{2f}$ rounds are required for a valid chain to grow by k blocks. Additionally, note that, for $f > 0.5$ the number of rounds to generate k blocks becomes less than k, which is not possible because multiple blocks in the same round will only increase forks, not the chain length. That necessarily means any $f > 0.5$ will not improve the chain growth, instead only increase the fork rate. That is why we should always consider $f \leq 0.5$.

A corollary to chain growth lemma(Lemma 5), Lemma 3 and Lemma 6 is the following theorem.

Theorem 2 (chain-growth). *In a typical execution, the chain growth property holds with parameters $\tau = (1-\epsilon)f$, $\sigma = 2f$ and $r_g > \lambda$.*

The above theorem provides an upper bound as well as a lower bound on the total number of blocks added to a chain $\mathcal{C}$ given a sequence of rounds S with a length $s > r_g$. For s rounds, the number of blocks x added to the chain $\mathcal{C}$ is upper bounded by σs and lower bounded by τs.

5.3 Common Prefix Property

Here we prove that honest parties eventually agree on a common chain in our ET consensus with z-test protocol. The main difference from bitcoin backbone analysis [28] is the following: In bitcoin, the total number of blocks an adversary can produce is bounded (with some probability, of course). However, in our ET consensus with z-test protocol, the z-test only allows us to bound the number of blocks *per chain*. So an adversary could create a theoretically infinite number of chains and generate blocks on all of them. It turns out, though, that this per-chain restriction is actually pretty strong. Below we present the formal proof.

Lemma 7 (Common Prefix Lemma). *In a typical execution, for two chains $\mathcal{C}_1$ and $\mathcal{C}_2$ with $\mathtt{len}(\mathcal{C}_2) \geq \mathtt{len}(\mathcal{C}_1)$, if $\mathcal{C}_1$ is adopted by an honest party at round r, and $\mathcal{C}_2$ is either adopted by an honest party or broadcasted by an honest party at round r, then $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \preceq \mathcal{C}_1$, for all $k \geq \max(2\lambda f, 4)$.*

Proof. Let us assume, for contradiction, there exists a $k > 2\lambda f$ such that $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$ or $\mathcal{C}_2^{\lceil k} \not\preceq \mathcal{C}_1$. Suppose, B^* be the last block on the common prefix of $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathtt{id}(B^*)$ is honest. Let us denote $\mathtt{round}(B^*) = r^*$. Note, B^* can be genesis block, in which case $r^* = 0$.

Now, we define $S = \{i : r^* < i < r\}$. Suppose, $B_m, B_{m+1}, \cdots, B_{m+k'-1}$ are k' consecutive blocks of the chain $\mathcal{C}_1$, where B_m is the next block after $B*$ and $B_{m+k'-1}$ is the last block of $\mathcal{C}_1$. Clearly, $k' \geq k \geq \max(2\lambda f, 4)$ and we can apply Lemma 6. This implies, $|[\mathtt{round}(B_m), \mathtt{round}(B_{m+k'-1})]| \geq \frac{k'}{2f} \geq \lambda$. We also know, $S \supseteq [\mathtt{round}(B_m), \mathtt{round}(B_{m+k'-1})]$. Hence, $|S| \geq \lambda$ (i.e., the execution during S is a typical execution with overwhelming probability) and Lemma 3 applies for the set of rounds S.

For a uniquely successful round $u \in S$, let j_u be the position at which the uniquely successful honest party created the block. J be the set of positions at which honest parties created the blocks on uniquely successful rounds. $J = \{j_u : u \in S, HON_u^1 = 1\}$. Suppose the maximum value of the set J is $\max(J)$. Then, $\mathtt{len}(\mathcal{C}_1) \geq \max(J)$, since $\mathcal{C}_1$ is adopted by an honest party at round r, by which the honest party has already received a chain of length $\max(J)$.

Since, $\mathtt{len}(\mathcal{C}_2) \geq \mathtt{len}(\mathcal{C}_1)$, j^{th} block exists in both the chains $\mathcal{C}_1$ and $\mathcal{C}_2$ for all $j \in J$. We denote such blocks by $B_{1,j}$ and $B_{2,j}$ respectively. Now, we want to claim for all $j \in J$ at least one of the players between $\mathtt{id}(B_{1,j})$ and $\mathtt{id}(B_{1,j})$ is corrupted. By Lemma 4, if both $\mathtt{id}(B_{1,j})$ and $\mathtt{id}(B_{1,j})$ are honest then we must have $B_{1,j} = B_{2,j}$. Cryptographic strength (collision resistance) of the hash function implies $B_j = B_{1,j} = B_{2,j}$ belongs to the common prefix of chains $\mathcal{C}_1$ and $\mathcal{C}_2$. However, we also know $\mathtt{round}(B_j) > \mathtt{round}(B^*)$ and B^* is the last block in the common prefix such that $\mathtt{id}(B^*)$ is honest. This implies a contradiction.

Now, we have established the fact that for all $j \in J$ at least one of the players between $\mathtt{id}(B_{1,j})$ and $\mathtt{id}(B_{2,j})$ is corrupted. Hence, total number of blocks B such that $\mathtt{id}(B)$ is corrupted, $B \in \mathcal{C}_1 \cup \mathcal{C}_2$ and $\mathtt{round}(B) \in S$ must be more than or equal to size of set J. Hence,

$$\begin{aligned} ADV_{\mathcal{C}_1}(S) + ADV_{\mathcal{C}_2}(S) &\geq |\{B : B \in \mathcal{C}_1 \cup \mathcal{C}_2 \text{ and } \mathtt{round}(B) \in S\}| \\ &\geq |J| = HON^1(S). \end{aligned}$$

However, for a typical execution with $|S| \geq \lambda$, by Lemma 3 we have

$$ADV_{\mathcal{C}_1}(S), ADV_{\mathcal{C}_2}(S) < \frac{HON^1(S)}{2}.$$

Hence, contradiction. Therefore, we can say that for all $k > 2\lambda f$, it holds that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$ and $\mathcal{C}_2^{\lceil k} \preceq \mathcal{C}_1$. □

Intuitively, if $\mathcal{C}_1^{\lceil k} \not\preceq \mathcal{C}_2$ or $\mathcal{C}_2^{\lceil k} \not\preceq \mathcal{C}_1$, the number of adversarial blocks for both the chains combined is more than the total number of honest blocks, for the parts of the chains where they don't have a common honest block. And that is not possible for a typical execution, because the number of adversarial blocks for a chain $\mathcal{C}$ during a sequence of rounds S is limited by $ADV_{\mathcal{C}}(S) < \frac{HON^1(S)}{2}$.

Common Prefix Lemma shows that the honest parties eventually agree on a common chain. Once a transaction is included in a block B, the transaction becomes irreversible once honest parties have mined enough number of blocks extending after B. The common prefix lemma directly implies the following security theorem about the common prefix property.

Theorem 3 (Common Prefix). *In a typical execution the common prefix property holds with parameter* $\ell_{cf} \geq \max(2\lambda f, 4)$.

5.4 Chain Quality Property

Now we want to prove the property that at least a constant fraction of blocks are added by honest parties in a chain $\mathcal{C}$ that is adopted by an honest party. That eventually ensures, because of common prefix property, that the common chain agreed on by the honest parties has at least a constant fraction of honest blocks.

Theorem 4 (Chain Quality). *In a typical execution, the chain quality property holds with parameters* $\ell_q \geq \max(2\lambda f, 4)$ *and* $\mu = 1 - \frac{(1+\epsilon')t}{(n-t)(1-f)(1-\epsilon)}$ *for any chain adopted by any honest party.*

Proof. Let us consider a chain $\mathcal{C}$, which has been adopted by an honest party P at round r, such that $\texttt{len}(\mathcal{C}) > \ell_q$. Suppose $\mathcal{C}$ consists of sequence of blocks $(B_1, B_2, \ldots, B_{\texttt{len}(\mathcal{C})})$ and $(B_u, B_{u+1}, \ldots, B_{u+\ell_q-1})$ is an arbitrary ℓ_q length subsequence of $\mathcal{C}$, such that $\ell_q \geq \max(2\lambda f, 4)$.

Let $(B_{u'}, B_{u'+1} \ldots, B_{u'+L-1})$ be the shortest subsequence of $\mathcal{C}$ containing $(B_u,\, B_{u+1}, \ldots, B_{u+\ell_q-1})$ (i.e. $u' \leq u$ and $L \geq \ell_q$) such that:

1. $\texttt{id}(B_{u'})$ is honest
2. there exists an honest party which adopted the chain $(B_1, B_2, \ldots, B_{u'+L-1})$

Observe that B_1 is genesis block and $\texttt{id}(B_1)$ is honest by definition. We know that an honest party P adopted the chain $\mathcal{C}$ and $\texttt{len}(\mathcal{C}) > \ell_q$. Hence, the whole chain $\mathcal{C}$ trivially satisfies the above properties, except it might not the shortest one. This shows existence of the shortest subsequence $B_{u'}, B_{u'+1} \ldots, B_{u'+L-1}$. Suppose, $r_1 = \texttt{round}(B_{u'})$ and the earliest round at which the chain $(B_1, B_2, \ldots, B_{u'+L-1})$ got adopted by an honest party is r_2. Let S be the sequence of rounds defined as $S = \{r : r_1 \leq r < r_2\}$. Observe that

$$S \supseteq [\texttt{round}(B_{u'}), \texttt{round}(B_{u'+L-1})] \supseteq [\texttt{round}(B_u), \texttt{round}(B_{u+\ell_q-1})].$$

Hence, by Lemma 6, we have $|S| \geq \ell_q/2f \geq \lambda$ and the properties of typical execution are applicable (Lemma 3) for the set of rounds S.

Let x be the number of honest blocks in the ℓ_q length sequence. In other words $x = |\{B \in (B_u, B_{u+1}, \ldots, B_{u+\ell_q-1}) | \texttt{id}(B) \text{ is honest}\}|$. For contradiction, we assume the chain quality property does not hold for this ℓ_q length sequence of blocks $(B_u, B_{u+1}, \ldots, B_{u+\ell_q-1})$. Hence, $x < \mu\ell_q \leq \mu L$.

As the chain $(B_1, B_2, \ldots, B_{u'+L-1})$ got adopted by an honest party in round r_2; for all $i \in [u', u'+L-1]$ we have $\texttt{round}(B_i) \in S$. As $[u, u+\ell_q-1] \subseteq [u', u'+L-1]$, from our contradiction assumption we have

$$\begin{aligned} ADV_{\mathcal{C}}(S) &\geq |\{B \in (B_u, B_{u+1}, \ldots, B_{u+\ell_q-1}) | \texttt{id}(B) \text{ is corrupted}\}| \\ &= L - x > (1-\mu)L \end{aligned} \tag{4}$$

Now, Lemma 5 implies $u' + L - 1 \geq u' + HON^{\geq 1}(S)$ or equivalently $L > HON^{\geq 1}(S)$. Hence inequality (4) can be rewritten as $ADV_{\mathcal{C}}(S) > (1-\mu)HON^{\geq 1}(S)$. As we have seen before, $|S| \geq \lambda$. Hence, by Lemma 3

$$(1-\mu)HON^{\geq 1}(S) = \frac{(1+\epsilon')t}{(n-t)(1-f)(1-\epsilon)} HON^{\geq 1}(S) \ > ADV_{\mathcal{C}}(S)$$

Therefore we have our desired contradiction $ADV_{\mathcal{C}}(S) > ADV_{\mathcal{C}}(S)$. □

The chain quality property guarantees that there will be at least $\mu\ell_q$ honest blocks given a chain of length ℓ_q. For example, when 20% of the miners are dishonest, $\epsilon' = \epsilon = 0.2$, and $f = 0.2$, we have $\mu = 0.53$—which means at least 53% blocks in the chain are honest. We refer to Table 2 for more examples.

6 Discussion and Practical Application

6.1 Parameter Choices

We want to set the z-test parameter ϵ' in such a way that an honest block is excluded from a chain only with negligible probability. We therefore recommend setting $\epsilon = \epsilon'$. In Table 2 we show some examples with possible values of ϵ, f, δ and how the parameters $\tau, \sigma, \ell_{cf}, \mu$ corresponding to the security properties(namely, chain growth, common prefix and chain quality) vary. Table 2 shows that we can vary $(f + \epsilon)$ up to 1, when $\delta = 1$ (which means all the protocol parties are honest). For $\epsilon = 0.2$ and $f = 0.2$, δ can be as low as 0.75, which means the protocol can tolerate up to 20% dishonest protocol parties. For small values of ϵ and f, ET consensus with z-test can tolerate up to 33% dishonest parties.

6.2 Implications of *z*-Test Security

In addition to lending evidence to support that the actual PoET protocol (and other similar protocols like proof of luck) is resilient to the compromise of some TEEs, we show a pretty surprising fact: basic proof of work consensus with a *z*-test *but no actual proofs of work, just "promises" from users* still remains secure with an honest majority assumption! Table 3 shows that our ET protocol without TEEs is not terribly worse (in terms of security) than ET consensus with trusted timer (and, similarly, bitcoin in the bitcoin backbone protocol).

While these numbers are not incredibly tight, the δ factors indicate that our proofs hold even when the number of adversarial parties is a (relatively large)

Table 2. How the security property parameters $(\tau, \sigma, r_g, \ell_{cf}, \ell_q, \mu)$ of ET consensus with z-test corresponding to chain growth, common prefix and chain quality properties vary based on the protocol parameters $\delta, \epsilon, f, \lambda$. The first column presents the values of ϵ and f defined in the honest majority assumption, the second column the minimum δ (accurate up to two decimal places) to satisfy the honest majority assumption; the third, fourth, fifth, sixth, seventh, and eighth columns are τ, σ, r_g, ℓ_{cf}, ℓ_q and μ respectively as described in Sect. 5. For all the cases, we use $\epsilon' = \epsilon$, and $\lambda \gg 4$. Note that f in our case is actually derived from p, however, to be comparable with similar works [7,28,29] we use f in the table.

Protocol parameters	δ	τ	σ	r_g	ℓ_{cf}	ℓ_q	μ
$\epsilon = 0.05, f = 0.05$	0.58	0.0475	0.1	λ	0.1λ	0.1λ	0.51
$\epsilon = 0.1, f = 0.1$	0.64	0.09	0.2	λ	0.2λ	0.2λ	0.51
$\epsilon = 0.2, f = 0.2$	0.75	0.16	0.4	λ	0.4λ	0.4λ	0.53
$\epsilon = 0.3, f = 0.3$	0.85	0.21	0.6	λ	0.6λ	0.6λ	0.60
$\epsilon = 0.4, f = 0.3$	0.88	0.18	0.6	λ	0.6λ	0.6λ	0.6
$\epsilon = 0.4, f = 0.4$	0.93	0.24	0.8	λ	0.8λ	0.8λ	0.72
$\epsilon = 0.5, f = 0.4$	0.95	0.2	0.8	λ	0.8λ	0.8λ	0.65
$\epsilon = 0.6, f = 0.4$	0.96	0.16	0.8	λ	0.8λ	0.8λ	0.73
$\epsilon = 0.5, f = 0.5$	1	0.25	1	λ	λ	λ	1

Table 3. Relationship between f and minimum δ in ET consensus with trusted timer, ET consensus with z-test. The f and the corresponding minimum δ values are exactly same for Bitcoin consensus and ET with trusted timer, and therefore, we do not include a separate column for Bitcoin in the table.

f	ET with trusted timer $\delta_{\min}$	ET with z-test $\delta_{\min}$
0.05	0.3	0.58
0.1	0.6	0.64

constant fraction (up to 33%) of the total number of players. This indicates that current TEE-based consensus systems like PoET that are used "in the wild" are, at least in theory, secure, although we would need to change the z-test in PoET in order for our proofs to apply.

Although the security proofs of ET consensus with z-test hold without any TEE assumptions, as long as the honest majority assumption holds, we recommend using the protocol in combination with TEE (e.g., Intel SGX) to ensure only a small number of malicious participants.

6.3 Performance Improvement

Even though, in our protocol description, we make ET_{ztest} wait on the TEE to generate a block, the security analysis does not depend on that. And therefore, a player can just query the $WaitTime$ and $WaitCert$ from the TEE, and still, all the security properties will hold. This can be very useful in practice, because if each round is small enough querying the TEE every round can be really inefficient.

6.4 Applications of Our Results

In [28], the authors show that the bitcoin backbone protocol almost immediately implies a Byzantine fault tolerant consensus protocol and a public ledger. The same results apply to our protocols, so we omit the full proofs and descriptions here. An inquisitive reader can refer to Sects. 5 and 6 of [28].

Acknowledgment. We thank the anonymous reviewers for their helpful comments. We thank Dan Middleton for the useful discussions.

Appendix: Security of ET Consensus with Trusted Timer

Assuming that the hash function and the signature scheme are cryptographically secure through our *Cert* () functionality, and assuming integrity of the TEE, we can prove that the security properties of our ET consensus protocol with trusted timer are exactly same as that of Bitcoin. This fact is a direct implication of Lemma 1, and the security proofs are extremely similar to that of Bitcoin. Here we skip the proofs and present the key security properties.

All of the security guarantees hold if there are enough honest parties in the system, where the exact amount that is "enough" depends on other parameters of the system. Below, we formally state the honest majority assumption.

Definition 7 (Honest Majority Assumption). *Suppose n is the total number of parties, and out of them t parties are corrupted. If δ is the advantage of honest parties, then we require that $t < (1-\delta)(n-t)$, where $3f + 3\epsilon < \delta \leq 1$, where ϵ is a positive fraction (used in various concentration bounds) and f is the probability that at least one honest party creates a block at a given round.*

For a security parameter λ, total number of parties $n \in poly(\lambda)$, and with the above honest majority assumption the following security theorems can be derived about our ET consensus protocol with trusted timer.

Theorem 5 (chain-growth). *The chain growth property holds with parameters $\tau = (1-\epsilon)f$, $\sigma = 2f$ and $r_g > \lambda$ with overwhelming probability.*

Theorem 6 (Common Prefix). *The common prefix property holds with parameter $\ell_{cf} \geq \max(2\lambda f, 4)$ with overwhelming probability.*

Theorem 7 (Chain Quality). *With overwhelming probability the chain quality property holds with parameters $\ell_q \geq \max(2\lambda f, 4)$ and $\mu = 1 - (1+\frac{\delta}{2})\frac{t}{n-t} - \frac{\epsilon}{1-\epsilon} > 1 - (1+\frac{\delta}{2})\frac{t}{n-t} - \frac{\delta}{2}$ for any chain adopted by any honest party.*

References

1. Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption
2. Introduction to hyperledger sawtooth. https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html
3. Poet 1.0 specification. https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html
4. Trusted time and monotonic counters with intel software guard extensions platform services. https://software.intel.com/sites/default/files/managed/1b/a2/Intel-SGX-Platform-Services.pdf
5. Andreina, S., Bohli, J.-M., Karame, G.O., Li, W., Marson, G.A.: Pots - a secure proof of tee-stake for permissionless blockchains. Cryptology ePrint Archive, Report 2018/1135 (2018). https://eprint.iacr.org/2018/1135
6. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, p. 30. ACM (2018)
7. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: a composable treatment. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 324–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_11
8. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
9. Benet, J., Greco, N.: Filecoin: a decentralized storage network. Protocol Labs (2018)
10. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 142–157. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_10
11. Bessani, A., Sousa, J., Alchieri, E.E.: State machine replication for the masses with BFT-SMART. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 355–362. IEEE (2014)
12. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 757–788. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_25
13. Bowman, M., Das, D., Mandal, A., Montgomery, H.: On elapsed time consensus protocols. https://eprint.iacr.org/2021/086.pdf
14. Brown-Cohen, J., Narayanan, A., Psomas, C.-A., Weinberg, S.M.: Formal barriers to longest-chain proof-of-stake protocols. CoRR, abs/1809.06528 (2018)
15. Buterin, V., Griffith, V.: Casper the friendly finality gadget. CoRR, abs/1710.09437 (2017)
16. Cachin, C.: Architecture of the hyperledger blockchain fabric. In: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, vol. 310 (2016)
17. Cachin, C., Vukolić, M.: Blockchains consensus protocols in the wild. arXiv preprint arXiv:1707.01873 (2017)
18. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI, vol. 99, pp. 173–186 (1999)

19. Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: On security analysis of proof-of-elapsed-time (PoET). In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 282–297. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_19
20. Cohen, B., Pietrzak, K.: The chia network blockchain (2019)
21. Corso, A.: Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework (2019)
22. Daian, P., Pass, R., Shi, E.: Snow white: provably secure proofs of stake. Technical report, Cryptology ePrint Archive, Report 2016/919, 2016 (2016)
23. Dang, H., Dinh, T.T.A., Loghin, D., Chang, E.-C., Lin, Q., Ooi, B.C.: Towards scaling blockchain systems via sharding. In: Proceedings of the 2019 International Conference on Management of Data, pp. 123–140 (2019)
24. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros Praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 66–98. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_3
25. Dolev, D., Reischuk, R.: Bounds on information exchange for Byzantine agreement. J. ACM **32**(1), 191–204 (1985)
26. Dziembowski, S., Faust, S., Kolmogorov, V., Pietrzak, K.: Proofs of space. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 585–605. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_29
27. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-ng: a scalable blockchain protocol. In: NSDI, pp. 45–59 (2016)
28. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
29. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_10
30. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16. ACM (2016)
31. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
32. Kocher, P., et al.: Spectre attacks: exploiting speculative execution. In: 40th IEEE Symposium on Security and Privacy (S&P 2019) (2019)
33. Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 279–296 (2016)
34. Koruyeh, E.M., Khasawneh, K.N., Song, C., Abu-Ghazaleh, N.: Spectre returns! Speculation attacks using the return stack buffer. In: 12th USENIX Workshop on Offensive Technologies (WOOT 2018) (2018)
35. Kwon, J.: Tendermint: consensus without mining. Draft v. 0.6, fall (2014)
36. Lipp, M., et al.: Meltdown: reading kernel memory from user space. In: 27th USENIX Security Symposium (USENIX Security 2018), pp. 973–990 (2018)

37. Liu, J., Li, W., Karame, G.O., Asokan, N.: Scalable byzantine consensus via hardware-assisted secret sharing. IEEE Trans. Comput. **68**(1), 139–151 (2019)
38. Madison, J.: Federalist no. 51. The Federalist Papers (1788)
39. Mazieres, D.: The stellar consensus protocol: a federated model for internet-level consensus. Stellar Development Foundation (2015)
40. Milutinovic, M., He, W., Wu, H., Kanwal, M.: Proof of luck: an efficient blockchain consensus protocol. In: Proceedings of the 1st Workshop on System Software for Trusted Execution, p. 2. ACM (2016)
41. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf
42. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: generalizing selfish mining and combining with an eclipse attack. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 305–320. IEEE (2016)
43. Pass, R., Shi, E.: Fruitchains: a fair blockchain. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 315–324. ACM (2017)
44. Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 380–409. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_14
45. Pass, R., Shi, E.: Thunderella: blockchains with optimistic instant confirmation. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 3–33. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_1
46. Santos Veronese, G., Correia, M., Neves Bessani, A., Lung, L.C., Verissimo, P.: Efficient Byzantine fault-tolerance. IEEE Trans. Comput. **62**, 16–30 (2013)
47. Shi, Z., Zhou, H., Hu, Y., Jayachander, S., de Laat, C., Zhao, Z.: Operating permissioned blockchain in clouds: a performance study of hyperledger sawtooth. In: 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 50–57. IEEE (2019)
48. Tamer Özsu, M., Valduriez, P.: Correction to: principles of distributed database systems. In: Principles of Distributed Database Systems, pp. C1–C2. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-26253-2_13
49. Van Bulck, J., et al.: Foreshadow: extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 991–1008 (2018)
50. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_9
51. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, pp. 347–356. ACM (2019)