Fairledger: a Fair Proof-of-Sequential-Work based Lightweight Distributed Ledger for IoT Networks

Ronghua Xu, Yu Chen
Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902, USA
{rxu22, ychen}@binghamton.edu

2022 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-6104-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/BLOCKCHAIN55522.2022.00055

***Abstract*—Blockchain has been recognized as a promising solution to construct a tamper-proof and trust-free decentralized framework for Internet of Things (IoT) systems. However, directly applying cryptocurrency-oriented blockchains in IoT networks still meets tremendous limitations. Proof-of-Work (PoW) consensus protocol enables a blockchain to achieve pseudonymity, scalability and probabilistic finality in an asynchronous and open-access network environment. The compute-intensive PoW favors nodes possessing more computing power, but fairness is an important requirement in highly heterogeneous IoT networks. A blockchain designed for IoT edge environments must consider devices with various constraints on computation power. This paper proposes Fairledger, a fair Proof-of-Sequential-Work (PoSW) based lightweight distributed ledger for small-scale, permissioned IoT networks. By combining efficient verifiable delay function (eVDF) and Proof-of-Credit (PoC) puzzle, PoSW consensus protocol guarantees fairness by requiring all the miners perform a fixed sequential computing steps to represent PoW for block generation despite their hardware resources. Due to the virtual mining manner of PoSW, Fairledger demonstrates computing efficiency compared to traditional PoW blockchains. In addition, Fairledger is less susceptible to "long-range" and "nothing-at-stake" attacks than Proof-of-Stake (PoS) protocols are. A proof-of-concept prototype is implemented, and the experimental results verify the feasibility of running Fairledger in a physical IoT network with higher throughput, less computation and communication cost, and better security guarantees.**

***Index Terms*—Blockchain, Fairness, Security, Proof-of-Sequential-Work (PoSW) Consensus, Verifiable Delay Functions (VDF), Internet-of-Things (IoT).**

I. Introduction

Thanks to the rapid advancements in artificial intelligence (AI) based on the fusion of Big Data and Internet of Things (IoT) technologies, the concept of Smart Cities becomes realistic to provide seamless, intelligent, and safe services for communities [29]. With an ever-increasing presence of IoT-based smart applications and their ubiquitous visibility from the Internet, the highly connected smart IoT devices generate a huge volume of transaction data and incur more concerns on performance, security, and privacy [9], [12]. The heterogeneity of IoT networks necessitates a scalable, flexible, and lightweight system architecture, which supports fast development, easy deployment, and secure data sharing among multiple fragmented service domains [28]. However, most state-of-the-art applications for smart cities heavily rely on a centralized authority, which is vulnerable to be a performance bottleneck and single point of failure, and faces heterogeneity and scalability challenges with wide adoption of IoT devices [4].

Blockchain, which acts as the underlying technology of cryptocurrencies like Bitcoin [21], has demonstrated great potential to revolutionize traditional business models and communication infrastructure [15]. In general, blockchain systems rely on a Peer-to-Peer (P2P) networking architecture for messages and data propagation. All participants (miners or validators) cooperatively execute a cryptographic consensus protocol to records blocks on a totally-ordered distributed ledger. Therefore, participants maintain a transparent, immutable, and auditable distributed ledger, as opposed to establishing trust through a centralized third-party authority. Thanks to attractive features like decentralization, immutability and auditability, blockchain is promising to construct a tamper-proof and trust-free framework for IoT systems.

However, directly applying cryptocurrency-oriented blockchain schemes in IoT networks still meets tremendous limitations [4], [10]. The existing permissionless blockchains (e.g., Bitcoin or Ethereum) uses compute-intensive Proof-of-Work (PoW) algorithms for block generation, which is not affordable to resource-constrained IoT devices. While byzantine fault tolerance (PBFT) protocol adopted by permissioned blockchains like Hyperledger demonstrates low energy consumption and high throughout, it allows limited scalability and incurs high communication complexity [8]. To reduce energy waste by PoW, Proof-of-Stake (PoS) [17] leverages the coin stakes of miners to mimic a random lottery for block generation. Such a process of "virtual mining" requires minimal computational resource for miners. As an adversary can easily use the same stake to grind many blocks without any computing efforts on solving the PoS puzzle problem, PoS protocols are vulnerable to cost-less simulation attacks [11], like long-range attacks and nothing-at-stake attacks.

To address aforementioned issues as cooperating conventional blockchains into IoT scenarios, this paper proposes Fairledger, a fair Proof-of-Sequential-Work (PoSW) based lightweight distributed ledger for small scale permissioned IoT Networks. The PoSW leverages an efficient verifiable delay function (eVDF) such that a validator can get an unique Fiat-Shamir heuristic proof until it performs a fixed number of repeated squaring modular exponentiations in a group of unknown order. Then, a Proof-of-Credit (PoC) algorithm simulates a lottery process based on the eVDF proofs such that the probability of winning a block proposal is proportional to credit stakes of the validators. Finally, a voting-based chain finality (VCF) process makes an agreement on the epoch

978-1-6654-6104-7/22/$31.00 ©2022 IEEE
DOI 10.1109/Blockchain55522.2022.00055

checkpoint block to solve forks and finalize blocks of the main chain on the distributed ledger.

The existing hybrid VDF-Nakamoto consensus protocols [11], [14], [16], [19], [23], [30] require each miner take a random and unpredictable numbers of sequential steps to calculate the VDF proofs, and then the miner who fist publishes a valid proof that can solve the pre-defined puzzle problem becomes the leader. However, such a competitive manner allows miners with fast single core processors to gain extra benefits. Unlike aforementioned solutions that can only guarantee a relative μ-fairness [14], Fairledger achieves an idea 1-fairness such that all participants consume the same amount of computation power to mine a new block regardless their hardware resource. In addition, PoC only needs proofs of eVDF and credit stakes of current committee members to decide if miners are qualified to generate new blocks. Therefore, it can avoid energy waste by brute-force querying hash in conventional PoW blockchains. In sum, Fairledger is computing-efficient and hardware independent, and it is suitable for resource-constrained IoT devices with heterogeneous platforms.

For security properties, the sequentiality of eVDF can prevent dishonest validators from using parallel computing scheme to speed up proof calculation, and it also mitigates long-range attacks and nothing-at-stake attacks in existing PoS style blockchains [17], [27]. In contrast to existing VDF-Nakamoto consensus protocols that rely on the "longest chain" rule to achieve a probabilistic finality, Fairledger requires a small committee to execute VCF to guarantee deterministic finality. Therefore, it is promising to solve fork issues and prevent against block withholding attacks.

In summary, this paper makes the following contributions:

1) A complete architecture of IoT blockchain called Fairledger is provided along with a detailed explanation of the key components and work flows;
2) A novel PoSW consensus protocol is introduced and formalized, which is a composition of eVDF construction with PoC-based block generation mechanism;
3) The security properties of Fairledger are described, and PoSW consensus protocol is briefly analyzed concerning several possible attacks; and
4) A proof-of-concept prototype is implemented and performance of running Fairledger is evaluated on a small group of Raspberry Pis. The numerical results show that Fairledger only incurs limited latency and communication overhead as scaling up committee size and system transactions throughput.

The remainder of this paper is organized as follows: Section II introduces background knowledge of VDFs and reviews existing VDF-based consensus protocols. Section III introduces the design rational and system architecture of Fairledger. A novel PoSW block proposal mechanism is explained in Section IV. Section V is the security analysis and Section VI presents the experimental evaluation on the prototype. Finally, Section VII concludes this paper with some discussions.

II. State of The Art and Related Work

A. Verifiable Delay Functions

The problem of building a Verifiable Delay Function (VDF) was formalized in 2018 [6]. A VDF $f : X \rightarrow Y$ requires a prescribed number of sequential steps to evaluate an input $x \in X$, independently of the parallel computing manners exploited by an adversary, and such that an unique output $y \in Y$ can be efficiently and publicly verified by a proof π. A VDF includes three basic algorithms: $Setup$, $Eval$ and $Verify$.

- $Setup(\lambda, T) \rightarrow$**PP** is the initialization procedure. It takes security parameter λ and delay parameter T and produces public parameters **PP** consisting of evaluation and verification key pairs (ek, vk) and other information for $Eval$ and $Verify$ procedures.
- $Eval($**PP**$, x, T) \rightarrow (y, \pi)$ is the evaluation procedure where all honest parties can compute (y, π) in time T of sequential steps. However, no parallel computation with a polynomial number of processors can calculate y in significant fewer steps.
- $Verify($**PP**$, x, y, T, \pi) \rightarrow (True, False)$ is a deterministic algorithm, which verifies that y is the unique correct output for x based on proof π in time $polylog(T)$.

A VDF must satisfy three security properties: *Sequentiality*, *Correctness*, and *Soundness*. Sequentiality guarantees that using parallelism to correctly evaluate VDF in time less than T sequential steps is almost impossible. Correctness ensures that any honest output of evaluation can pass the verification, while soundness states that a dishonest evaluation result is always rejected by verification.

To solve time-lock puzzles [24], RSW-VDF aims to provide a timed-release cryptographic mechanism such that encryption can only be decrypted until a pre-determined amount of time has passed. Relying on a trusted setup to generate a classical Rivest–Shamir–Adleman (RSA) modules $N = pq$ with $\phi(N) = (p-1)(q-1)$, RSW-VDF performs T repeated modular squarings to calculate $y = x^{2^T} \bmod N$, while any party knowing $\phi(N)$ can quickly calculate $y = x^e \bmod N$ where $e = 2^T \bmod \phi(N)$. However, RSW-VDF is an interactive protocol that cannot support efficient and public verification without sharing $\phi(N)$. Sloth [18] uses chaining modular square roots to build a slow-timed hash function to enable trustworthy and non-interactive generation of public random numbers. Sloth guarantees time security given that modular square root extraction cannot be done faster than modular exponentiation in evaluation. However, sloth-VDF construction does not ensure asymptotically efficient verification.

Two recent practical VDF constructions, a simple VDF (sVDF) [22] and an efficient VDF (eVDF) [26], leverage exponentiation in a group of unknown order and Fiat-Shamir heuristic based on hash prime to ensure ε-evaluation time, sequentiality and uniqueness properties [7]. Both VDFs uses $Setup(\lambda, T)$ to generate public parameters **PP** $= (G, H, T)$ including a finite abelian group G of unknown order and an efficient hash function $H : X \rightarrow G$. The evaluation algorithm

of VDFs computes T modular squarings of $H(x)$ and outputs an evaluator $y \leftarrow H(x)^{2^T} \in G$. However, sVDF and eVDF rely on different public-coin succinct arguments for calculating proof π and verifying y. In proof generation, sVDF is more efficient by using $O(\sqrt{T})$ group multiplications than eVDF that needs $O(T)$. As a trade-off, overall proof π of sVDF contains $\log_2 T$ elements in G, while eVDF's proof π has a single element in G. Compared with sVDF that takes $2\log_2 T$ exponentiations in verification, eVDF outputs the short proof and its verification is faster by executing two exponentiation. A study compares performance between sVDF and eVDF in terms of computation and storage [5]. As sVDF needs the larger block size to record total proof of $\log_2 T$ elements, this is not suitable for blockchain which requires compact transaction and block data structure to ensure communication efficiency. This paper uses eVDF to construct VDF, which requires non-parallel computational delay in block generation yet efficient block verification.

B. VDF-based Consensus Protocols

The existing VDF-based consensus solutions aim at fairly mining for Nakamoto style probabilistic consensus protocols, like PoW and PoS. To enable a scalable and fair distributed randomness beacon (DRB) protocol, Sequential Proof-of-Work (SeqPoW), a hybrid VDF+PoW consensus algorithm, is introduced to satisfy both sequentiality and hardness [14]. SeqPoW can prevent against parallelizable hash querying of PoW by using VDF to solve a cryptographic puzzle, which needs a random and unpredictable number of sequential steps. Due to sequentiality and hardness of SeqPoW, using multiple processors do not have any advantage in solving a puzzle such that fairness is achieved. Similar to SeqPoW, a improved blockchain consensus protocol is proposed, which utilizes a distributed VDF (DVDF) to replace conventional difficulty adjustment algorithm (DAA) of PoW [30]. The sequentiality of DVDF allows miners to perform hash calculation within a relatively fixed time despite their considerable hashrate. As a result, the hashratre fluctuations of DDA is mitigated to achieve stable rate of block generation.

To design an energy efficient consensus protocol which has good resistance to the long range attacks, a proof-of-stake/proof-of-delay hybrid protocol [19] is proposed by incorporating VDF into a general PoS blockchain [17]. The verifiable random function (VRF) acts as a pseudo-random oracle to assign different verifiable delay puzzles to the miners for a new block generation. The lottery representation that takes a different number of sequential steps to solve the delay puzzle in proportion to the percentage of the miners' ownership of a cryptocurrency. Similar to [19], Proof of staked hardware (PoSH) [16], a concept of PoW/PoS/PoSeq hybrid protocol, is proposed to optimize energy utilization for Nakamoto-style blockchain. However, they do not provide details of protocol design and security analysis.

Another concurrent VDF-PoS protocol, called PoS with Arrow-of-Time PoSAT [11] provably achieve dynamic availability fully without additional trust assumptions for permissionless PoS blockchains. Due to unpredictability of random sequential steps of VDF computation based on the parent block till the threshold in proportion to coin stakes of the miners, an adversary cannot predict PoSAT lottery within the epoch of the c blocks generation. PoSAT provides a complete theoretical protocol design and security analysis for integrating VDF into existing PoS consensus protocols, however, implementation study is not mentioned to verify feasibility as applying PoSAT to practical IoT scenarios.

Given a round-robin committee selection mechanism based on a Trust Execution Environment (TEE) infrastructure, R3V [23] requires that selected stakeholders compete to generate blocks by solving VDF puzzles. R3V achieves to ensure fairness, communication efficiency and less susceptibility to long-range and grinding attacks in PoS blockchains, however, numerical results are not provided to verify performance.

III. Fairledger: Rationale and Architecture

A. System Model

System setting. Fairledger considers a permissioned network $\mathcal{N}$ that assumes the system administrator is a trustworthy oracle to register all nodes (participants or users) $u_i \in \mathcal{N}$. Given a trust Public Key Infrastructure (PKI) managed by the system administrator, a standard RSA scheme is adopted for key generation ($RSA.gen$) and digital signature ($RSA.sign$, $RSA.verify$), and a pre-defined collision-resistant hash function $\mathcal{H}(\cdot)$ is used to generates a hash string $h \in \{0,1\}^k$ with a length of k. Each node u_i has a key pairs $(sk_i, pk_i) \leftarrow RSA.gen(i)$ and is uniquely identified by its account address $a_i = \mathcal{H}(pk_i)$. In addition, each node u_i uses its credit stake c_i to join consensus protocol. Thus, each node has a tuple (pk_i, a_i, c_i) to represent its identity. Due to the permissioned system management, Fairledger is a partially decentralized blockchain.

Network model. Fairledger assumes a synchronous network environment, where message delivery for operations between nodes is bounded by a known finite time latency T_Δ given local processing time discrepancies and network delays. Thus, *Epoch* $sl_E = \{sl_1, sl2, ..., sl_t ..., sl_R\}$ is defined to model a set of sequential time slot sl_t in consensus rounds, where R value is the epoch size and $sl_t \geq T_\Delta$. Fairledger relies on a structured P2P network called Kademlia [20] for node discovery, such that each node only directly connect to a subset of nodes as its neighbours. All validators in a consensus committee are fully connected.

Adversary model. The adversary is adaptive such that it can control $m_i \in \mathcal{M} \subset \mathcal{N}$ at any time, and the fraction of compromised nodes is no more than $f = |\mathcal{M}|/|\mathcal{N}|$. Fairledger randomly selects a consensus committee called *Dynasty*, that includes *validator* $v_{i\in[1,K]} \in D$ where $K = |D|$. We assume the adversary is static in a consensus committee. Thus, the adversary can coordinate compromised nodes to disturb consensus protocol but cannot increase percentage of malicious validators in current D.

B. Architecture Overview

Figure 1 illustrates Fairledger system architecture including a tree style structure of distributed ledger $\mathcal{C}$ and

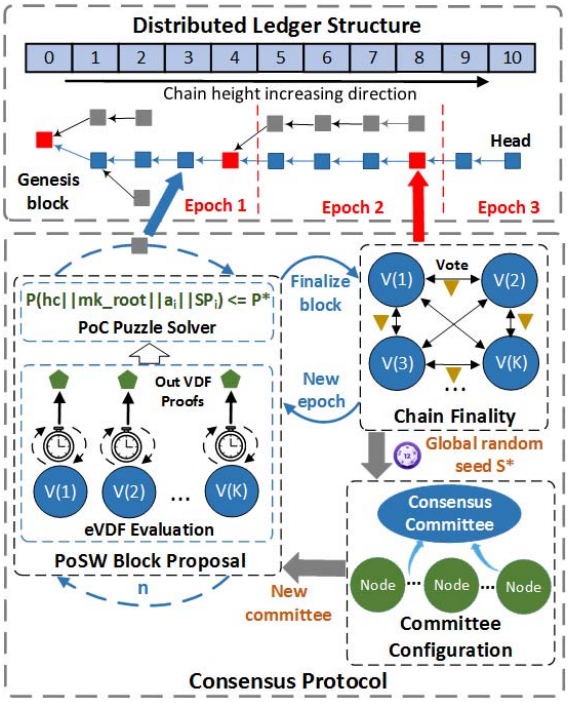

Fig. 1. The Fairledger System Architecture.

committee consensus protocol. Each node u_i can encapsulate byte strings $data \in \{0,1\}^*$ into a time stamped transaction $tx = \{tx\_hash, a_i, T_{stamp}, data, \sigma_i\}$, where $tx\_hash = \mathcal{H}(a_i, T_{stamp}, data)$ and σ_i is a digital signature $RSA.sign_{sk_i}(tx\_hash, pk_i, T_{stamp}, data)$. In current PoSW block proposal round within sl_t, a validators v_j that computes valid eVDF proofs SP_j to solve the PoC puzzle problem can publish a signed candidate block $B = (pre\_hash, h+1, mt\_root, tx\_list, SP_j, T_{stamp}, a_j, \sigma_j)$, where mt_root is a root hash of Merkle tree of valid transactions tx_list and h is the height of a head block $head(\mathcal{C})$.

The structure of ledger $\mathcal{C}$ is a partial order of blocks indexed by strictly sequential increasing h. For every epoch height $H_e = \lfloor \frac{h}{R} \rfloor$, a consensus committee uses a chain finality process to make an agreement on a *checkpoint* block, which is the current head block with $(h \mod R) = 0$. As the upper part of Fig. 1 shows, red blocks are checkpoint blocks, while blue ones are confirmed blocks that grow along the finalized chain path. The key components and work flows in consensus protocol are described as follows:

1. Committee Configuration: At the end of a dynasty lifetime, each $v_i \in D$ uses a pseudo random number generator to locally calculate a random seed S_i. By leveraging a randomness generation protocol that is based on Publicly Verifiable Secret Sharing (PVSS) [25], the current committee collaboratively forges a global epoch random seed S^* that is the sum of collected S_i. Then, all nodes use their ID profile and S^* to participate a new committee selection process based on a Verifiable Random Function (VRF) enabled cryptographic sortition [13]. Finally, the validators of a new committee establish a fully connected P2P consensus network. Furthermore, the system administrator uses S^* to generate a big prime N with λ length. The new committee configuration and public parameters (N, T and λ) are recorded into the first block of a new dynasty.

2. PoSW Block Proposal: Each validator maintains a locally transactions pool that is denoted as a time ordered list $TX = \{tx_i, tx_2, ..., tx_q\}$, where q is transaction pool size. The PoSW block proposal mechanism relies a hybrid VDF+PoC scheme consisting of two sub-protocols: eVDF evaluation and PoC puzzle verification. In a block proposal round, each $v_i \in D$ firstly executes $eVDF.EVal(\cdot)$ function based on inputs of mt_root of TX, hash string of head block hc and its account address a_i, and then an unique output tuple SP_i including evaluate l_i and proof π_i can be calculated after T sequential steps of squaring modular exponentiation on a big prime N (§IV-A). Then PoC solver sub-protocol decides if a valid SP_i can solve a PoC puzzle problem $P(\cdot)$ such that its owner v_i is qualified to propose a new block (§IV-B). Finally, a v_i with valid SP_i builds a new block and broadcasts it to all committee members. Each committee member accepts all valid blocks that satisfy $eVDF.Verify(\cdot)$ and PoC conditions P^* in the current block proposal round and appends them onto the head block of local chain. Similar to PoS that relies on a "longest chain" rule, the ledger extension in an epoch lifetime follows a "largest height of block" manner.

3. Chain Finality: At the end of an epoch, a voting-based algorithm commits checkpoint blocks and finalizes those already committed blocks on the main chain [27]. The chain finality ensures that only one path growing along with checkpoint blocks becomes the main chain, as blue arrows highlighted in Fig. 1. Therefore, the blocks generated in the new epoch are only extended on such an unique main chain. Compared to the longest chain rule used in Nakamoto blockchains, regularly resolving conflicting checkpoints can prevent against the forking chain problem and ensure a deterministic finality on history of the distributed ledger.

IV. PoSW Block Proposal Mechanism

A. eVDF Construction

Each validator $v_i \in D$ can execute $eVDF.Setup(T, N, \lambda)$ to construct an eVDF instance including i) a finite abelian group G of unknown order; ii) an efficient hash function $H : X \to G$; and iii) $H_{prime}(m) = next\_prime(H(m))$ that returns the closest prime numbers larger or equal to $H(m)$ for any $m \in \{0,1\}^*$ [26].

In the eVDF evaluation procedure, a validator v_i must solve the challenge $y = x^{2^T} \mod N$ to get an evaluator y given $x = H(m)$. This requires total T sequential steps of squaring modular exponentiation on N. Lines 3-5 in Algorithm 1 present the evaluation phase. The proof process requires v_i firstly compute $l = H_{prime}(x + y)$ and then calculate $\pi \leftarrow x^{\lfloor 2^T / l \rfloor} \mod N$. Finally, the pair $SP_i = (l, \pi)$ can be publicly uses as a Fiat-Shamir heuristic proof of evaluator y for verification. Lines 6-7 in Algorithm 1 present the proof calculation phase.

Algorithm 1 Evaluation and verification of eVDF.

1: **procedure**: eVDF.Eval(m, T, N)
2: $x \leftarrow H(m)$; $y \leftarrow x$
3: **for** $t \leftarrow 1$ to T **do**
4: $y \leftarrow y^2$ mod N
5: **end**
6: $l \leftarrow H_{prime}(x+y)$
7: $\pi \leftarrow x^{\lfloor 2^T/l \rfloor}$ mod N
8: **return** (l, π)
9: **procedure**: eVDF.Verify(m, T, N, (l, π))
10: $x \leftarrow H(m)$
11: $r \leftarrow 2^T$ mod l
12: $y \leftarrow \pi^l x^r$ mod N
13: **if** $l \neq H_{prime}(x+y)$ **then**
14: **return** *False*
15: **return** *True*

In the eVDF verification procedure, any validator $v_j \in D$ can use $SP_i = (l, \pi)$ to non-interactively check if validator v_i correctly calculates its evaluator $y = x^{2^T}$ mod N. As y can be locally recovered by calculating r=2^T mod l and then $y \leftarrow \pi^l x^r$ mod N, any party can compare l with output of $H_{prime}(x+y)$ to perform the verification on m and SP_i. This only needs total λ^4 time and is independent of T, therefore, the asymptotically efficient verification is achieved. Lines 10-15 in Algorithm 1 present the verification phase.

B. PoC-based Block Generation

Essentially, the PoC-based block proposal mechanism follows the principles of chain based PoS, and it simulates a virtual mining manner by pseudorandomly assigning block proposal rights to validators. The credit distribution of a dynasty is represented as $\mathcal{D} = \{p_1, p_2, ..., p_K\}$, where $p_i = \frac{c_i}{\sum_{j=1}^{K} c_j}$. For each time slot sl_t round, a random slot leader selection process called PoC puzzle solver determines whether a validator v_i with proof SP_i is allowed to propose a new block. Unlike PoW that utilizes a brute-force manner to query a nonce as the proof to meet the uniform target difficulty, our PoSW requires that each v_i takes the same sequential steps to calculate $SP_i = eVDF.EVal(hc||mt\_root||a_i||)$ as the proof for PoC puzzle problem. The PoC puzzle problem can be formally defined as follows:

***Definition** 1:* Proof-of-Credit Puzzle - Given an adjustable difficulty condition parameter ξ, the process of PoC puzzle solver aims to verify a solution string poc_proof proposed by v_j, which is calculated by taking ξ length lower bits of the hash string of $(hc||mt\_root||a_j||SP_j)$, is smaller than a target value generated by the difficulty condition $d_{cond}(\xi, p_j)$:

$$\mathcal{P}(\mathcal{H}((hc||mt\_root||a_j||SP_j)), \xi) \leq d_{cond}(\xi, p_j) \quad (1)$$

where function $\mathcal{P}(\cdot)$ outputs a poc_proof with lower ξ bits of the hashcode $\mathcal{H}(\cdot)$; and the difficulty condition function $d_{cond}(\cdot, \cdot)$ is denoted as:

$$d_{cond}(\xi, p_j) = (2^{\xi} - 1) \cdot p_j \quad (2)$$

where $d_{cond}(\xi, p_j) \in \{0, 1\}^{\xi}$.

Algorithm 2 The PoC block generation procedures.

1: **procedure**: mine_block(*TX*)
2: $hc \leftarrow \mathcal{H}(head(\mathcal{C}))$; $h \leftarrow head(\mathcal{C}).h + 1$
3: $mt\_root \leftarrow MTree(TX)$
4: $SP_i \leftarrow$ **eVDF.Eval**$((hc||mt\_root||a_i), T, N)$
5: $d_{cond} \leftarrow (2^{\xi} - 1)p_i$
6: *poc_proof* $\leftarrow \mathcal{P}(\mathcal{H}(hc||mt\_root||a_i||SP_i), \xi)$
7: **if** *poc_proof* $\leq d_{cond}$ **then**
8: *block* $\leftarrow (hc||mt\_root||TX||h||a_i||c_i||SP_i)$
9: $\sigma_i \leftarrow$ RSA.sign($block, sk_i$)
10: **return** $(block, \sigma_i)$
11: **end if**
12: **procedure**: verify_block(*block*, σ_j)
13: **if** RSA.verify($block, \sigma_j, pk_j$) $\neq True$ OR
14: $block.mt\_root \neq MTree(block.TX)$ **then**
15: **return** $False$
16: **end if**
17: $hc \leftarrow \mathcal{H}(head(\mathcal{C}))$
18: **if** $block.h \neq head(\mathcal{C}).h + 1$ OR
19: $block.hc \neq hc$ **then**
20: **return** $False$
21: **end if**
22: $m_j = (hc||block.mt\_root||a_j)$; $SP_j = block.SP$
23: $d_{cond} \leftarrow (2^{\xi} - 1)p_j$
24: *poc_proof* $\leftarrow \mathcal{P}(\mathcal{H}(hc||mt\_root||a_j||SP_j), \xi)$
25: **if** *poc_proof* $> d_{cond}$ OR
26: **eVDF.Verify**$(m_j, T, N, SP_j) \neq True$ **then**
27: **return** $False$
28: **end if**
29: **return** $True$

Given the above definitions, PoC-enabled block generation procedures are presented as pseudo-code in Algorithm 2. Given transactions pool TX and head block of ledger $\mathcal{C}$, each validator v_i executes the mining routine $mine\_block(\cdot)$ to probably get a candidate block based on its credit stake. Lines 2-3 prepare head information and mt_root, and then line 5 is the process of calculating eVDF proof SP. Lines 5-10 represent the procedure of PoC puzzle solver, which computes PoC solution poc_proof and publish a block singed by sk_i if poc_proof satisfies the difficulty condition d_{cond} based on p_i. Other validators $v_j \in D$ keep receiving blocks and execute $verify\_block(\cdot)$, which verifies blocks, and then add valid ones to their local ledger $\mathcal{C}$. The verification includes: i) validating the signature of a block along with mt_root of its saved transactions (lines 13-16); ii) checking if a block is directly linked to the head block with correct height (lines 17-21); iii) verifying eVDF proof SP and poc_proof of a block (lines 22-28). If all conditions are satisfied, v_j accepts a block B and updates $head(\mathcal{C}) = B$ accordingly. Otherwise, v_j discards all invalid blocks.

C. Ledger Extension Rules in an Epoch

The probability of a block B generated by v_i is proportional to its credit weight p_i, thus, the number of candidate blocks in a sl_t round is denoted as $b \in [0, K]$. To ensures the liveness such that there is at least one block can be mined in each round, the ledger extension rules in an epoch are described as follows:

1) $b = 1$: it is a basic scenario that only one candidate block B_{h+1} is proposed at height $h = head(\mathcal{C}).h$.

2) $b > 1$: it is a conflicting head block update scenario as multiple validators propose their valid blocks during current slot round. The ledger head update follows two sub rules:

a) $head(\mathcal{C})$ points to a block B_{h+1} whose sender has the highest credit c than other blocks' senders; or

b) if blocks are generated by validators who have the same highest credit, $head(\mathcal{C})$ points to a block B_{h+1} with the smallest poc_proof.

3) $b = 0$: if no block satisfies PoC conditions at the end of a slot round, block generation follows a spin manner. As validators of current committee can be sorted by account address, we can calculate $ind = height \pmod K$. Thus, a validator at rank ind becomes the leader to propose a candidate block in current round, and updating the head block follows the rule 1) $b = 1$.

V. Security Analysis

A. Security Properties

We assume that an adversary is subject to the usual cryptographic hardness assumptions, he/she is aware of neither the private keys of the honest nodes nor $\phi(N)$ of the big prime N. Fairledger utilizes PoSW consensus protocol to achieve security properties: fairness, liveness, and consistence.

Fairness: specifies that each validator is allowed to publish a new block only if he/she can compute an unique proof SP after a fixed T steps of non-parallelizable computation regardless of its financial stake or computation power. If an adversary exploits a parallel algorithm $\mathcal{A}$) by using at most $poly(\lambda)$ processors to speed up $eVDF.Eval(\cdot)$, the probability of calculating correct $\tilde{SP}$ less than time T is

$$Pr\begin{bmatrix} pp \leftarrow Setup(\lambda), \\ SP \leftarrow eVDF.Eval(pp, x), \\ \tilde{SP} \leftarrow \mathcal{A}(pp, x), \\ eVDF.Verify(pp, \tilde{SP}) = True \end{bmatrix} \leq negl(\lambda).$$

Therefore, sequentiality of eVDF evaluation process ensures the advantage of an adversary over the honest nodes is confined by a neglect function $negl(\lambda)$ even with a potential large number of parallel processors. Moreover, for an input $x \in X$, there is only exactly one output $y = x^{2^T} \pmod N$. Therefore, eVDF has uniqueness property, such that any proof pairs (l, π) that are generated from an estimator y can be accepted by the verification process.

Liveness: ensures that transactions submitted by the honest nodes are recorded into blocks and finalized on the distributed ledger after a sufficient amount of time defined by an epoch. In each round of block generation, a set of validators attempt to create new blocks, and only one block is accepted as the head block to extend the ledger. Therefore, a new block is always appended on the main chain in each sl_t round given the ledger extension rules, and fairledger achieves liveness.

Consistence: ensures the safety goal that all honest validators accept valid blocks and agree on the checkpoint blocks of the ledger. Thus, the probability of reverting the finalized blocks is negligible as the ledger continuously grows. Our fairledger relies on a deterministic chain finality to achieve consistence. Given assumption that an adversary can only control no more than f nodes if the total number of nodes satisfies $n \geq 3f + 1$. Therefore, the adversary has at most $m = 1/4$ chance per round to control the checkpoint voting process. As a result, the probability that an adversary controls n consecutive checkpoint is upper-bounded by $P[X \geq n] = \frac{1}{4^n} < 10^{-\tau}$. For $\tau = 6$, the adversary will control at most ten consecutive chain finality runs with the probability 10^{-6}.

B. Possible Attacks

Nothing-at-Stake Attack: As one of common attacks in PoS blockchains, the adversary can use the same stakes to easily mine many blocks on different forks without any resource cost, like computation or storage. As a result, attackers can extend multiple branches to maximize benefits. However, our PoSW consensus requires that each validator can mine a new block by performing a computation consisting of non-parallel sequential steps, it's difficult for a dishonest validator to generate many blocks within a block proposal round given the marginal performance gap between single cores.

Long-Range Attacks: Owing to the cost-less mining process in PoS blockchains, the adversary can create a fast growing branch starting from an earlier block (or even a genesis block) to overtake the longest main branch of the blockchain. The success of long-range Attacks leads to revert the history of public ledger. In our fairledger, the sequentiality of eVDF proof calculation and chain extension rules makes it expensive for the adversary to continuously propose valid blocks. Moreover periodically performing the deterministic chain finality on checkpoint blocks of an epoch makes the probability of successful long-range attack negligible.

Block Withholding Attacks: The adversary can mine blocks on its private branch of the ledger, then strategically publish later to gain extra rewards by reverting the main chain. In our fairledger, new blocks generation is divided into rounds, and mining blocks during current round are based on previous block. Therefore, an adversary can only launch a block-withholding attack only if he/she correctly calculates eVDF proofs and wins block proposal right in successive rounds. Nonetheless, the chances of successful attacks are low.

Content Grinding Attacks: In a content grinding attack, an adversary can parallel mine multiple blocks based on mt_root by using different sets and orders of transactions and then choose the most advantageous candidate block. To successfully launch such a attack, an adversary must literately compute eVDF proofs until a proof can solve the PoS puzzle problem before the end of a round. The sequential executions in eVDF evaluation imposes extra computation cost on attackers compared with PoS blockchains. However, an adversary with powerful single core may have advantage of winning a grinding attack, and we leave investigation and mitigation in future work.

VI. Implementation and Evaluation

A proof-of-concept prototype of Fairledger is implemented in Python. We use gmpy2 [2] to support multiple-precision

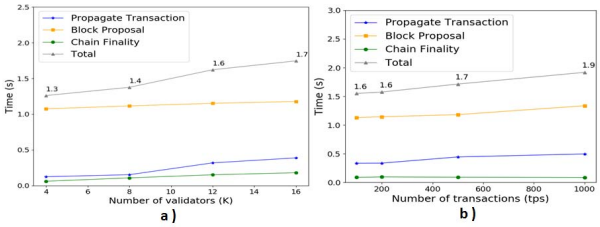

Fig. 2. Comparison of latency for an epoch cycle of PoSW consensus protocol as scaling: a) validators; b) tps.

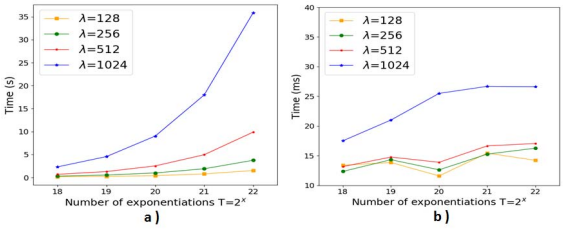

Fig. 3. Comparison of processing time on the validator as scaling parameters (λ, T): a) mine block; b) verify block.

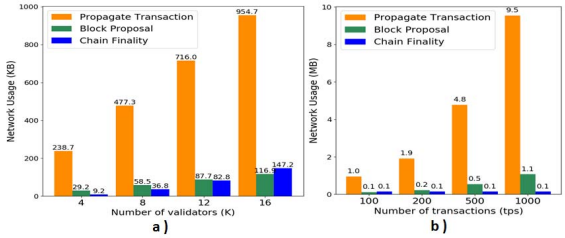

Fig. 4. Comparison of network usage as scaling: a) validators; b) tps

arithmetic operation on big integers and Flask [1] to develop networking and web service functions. All security functions like asymmetry cryptography and hash functions are developed by using standard python library cryptography [3].

The prototype is deployed on a physical network environment consisting of 16 validators that are deployed on 16 separate Raspberry Pi 4 (Rpi) devices with Quad core Cortex-A72 (ARM v8) CPU at 1.5GHZ CPU and 4 GB memory. While the system administrator is deployed on a desktop with Intel core i7-2600K (8 cores) CPU at 3.4GHZ and 8GB memory. All devices are connected through a local area network (LAN). We conduct 20 test runs for each test scenario and take the average as numerical results for performance evaluation.

A. End-to-End Latency

We let $\lambda = 256$ and $T = 2^{20}$ for eVDF functions and make system transaction throughput Th_S=100 Transactions Per Second (TPS). Figure 2a) presents the network latency for Fairledger to complete an entire round of consensus protocol given the number of validators K varying from 4 to 16. The latency includes the round trip time (RTT) and function processing time on the remote host. As propagate transaction and chain finality needs $\mathcal{O}(K)$ communication complexity for data broadcasting and verification, the delays of propagating transaction $\mathcal{T}_{pt}$ and chain finality $\mathcal{T}_{cf}$ are linear scale to committee size. The block proposal latency $\mathcal{T}_{bp}$ is dominated by eVDF evaluation process given a small consensus network, as a result, it demonstrates stable delays.

To observe the influence of numbers of transactions, we fix committee size K=16 while increasing Th_S. Figure 2b) shows the latency trends of the consensus protocol as scaling Th_S from 100 tps to 1000 tps. As larger Th_S means longer processing time for broadcasting transactions and calculating mt_root in a block generation, therefore, $\mathcal{T}_{pt}$ and $\mathcal{T}_{bp}$ are almost linear scale to Th_S. Fairledger leverages a small scale consensus committee to reduce delays of propagating transactions and blocks among validators. The latency of committing a new block on the ledger mainly depends on eVDF estimator and proof calculation.

B. Processing Time

To evaluate the impact of eVDF parameters (λ, T) on PoSW processing time on host machine, we let Th_S=100 tps and K=16 as scaling λ and T. Figure 3a) shows the processing time of mining a block by the validator as scaling T given different λ. As the most computational intensive process in mining a block, eVDF evaluation requires sequential T steps of m^2 such that mining time is linear scale to $\mathcal{O}(T)$. However, eVDF proof verification has $\mathcal{O}(\log_2 T)$ complexity, therefore, the validator takes negligible time of block verification compared to block mining, as Fig. 3b) shows. The security parameter λ influences the length of big prime N, so that the processing time increases when λ increases. However, the verify block process is still asymptotically efficient even with 1024 bit RSA groups.

C. Data Throughput

Given λ=256, $T = 2^{20}$ and $Th_S = 100$ tps, Fig. 4a demonstrates data transmission for individual stages of a consensus round as scaling up committee size. Fairledger uses json format for data transmission. Each transaction includes a 128 bytes data and has the size d_{tx}=430 Bytes, while a vote message has fixed size d_{vt}=589 Bytes. Total data transmission of propagating transactions denotes as $\mathcal{D}_{pt}$=$d_{tx} \times Th_S \times K$, and it linearly scales to K with fixed Th_S. For a chain finality round, broadcasting vote messages needs total data transmission $\mathcal{D}_{cf}$=$d_{vt} \times K^2$, which scales to K^2. Each block has the fixed header d_{header}=613 Bytes and only contains hash values of the transactions with size d_{tx_hash}=68 Bytes, and a block size is d_B=$d_{header} + d_{tx\_hash} \times Th_S$. Thus, data transmission of block proposal can be calculated as $\mathcal{D}_{bp}$=$d_B \times K$=$d_{header} \times K + d_{tx\_hash} \times Th_S \times K$, which is almost scale to K when Th_S is fixed.

We let K=16 to repeat test. Figure 4-b shows network usage as increasing Th_S. As the chain finality is independent of Th_S such that $\mathcal{D}_{cf}$ is stable with the fixed committee size. The $\mathcal{D}_{ct}$ depends on $Th_S \times d_{tx}$, and varying d_{tx} also influences network usage. However, the block size d_B is linearly scale to Th_S regarding the fixed size of d_{header} and d_{tx_hash}. Therefore, $\mathcal{D}_{bp}$ is independent to different transaction data and keeps low increasing rate.

Given results in Fig. 3b) and Fig. 4b), the total latency is calculated as $\mathcal{T}_M$=$\mathcal{T}_{pt} + \mathcal{T}_{bp} + \mathcal{T}_{cf}$, and then we can calculate

TABLE I
DATA THROUGHPUT VS. TRANSACTIONS.

Th_S **(TPS)**	100	200	500	1000
d_B **(KB)**	7.5	14.3	34.7	68.7
Th_D **(MB/s)**	0.8	1.4	3.2	5.7

data throughput $Th_D = \frac{\mathcal{D}_{pt}+\mathcal{D}_{bp}+\mathcal{D}_{cf}}{\mathcal{T}_M}$ (MB/s). Table I provides block size and data throughput with variant Th_S as committee size K=16. Given the maximum transactions list in Fairledger, increasing Th_S can improve capacity by recording data on the distributed ledger, and it implies a theoretical maximum data throughput of 5.7 MB/s as Th_S=1000 (tps), which can satisfy network bandwidth requirements of majority IoT networks.

VII. CONCLUSIONS

This paper presents a lightweight distributed ledger for IoT networks called Fairledger, which leverages a novel PoSW consensus protocol to achieve fairness, liveness, and consistency. Thanks to the sequentiality of eVDF proofing and efficiency of PoC puzzle solver, Fairledger is promising to prevent against cost-less simulation attacks [11] in existing PoS blockchains. The experimental results are based on a Fairledger in practical IoT scenarios.

While the prototype of Fairledger mainly demonstrates performance improvements, like high throughput, low latency and limited bandwidth usage, more investigation and test are necessary to evaluate the scalability and security of random committee election. Thus, our on-going efforts includes validating Fairledger in real-world IoT applications and evaluating overall performance and security based on various attack scenarios. Moreover, there are unanswered questions on incentives mechanism that motivates devices to devote their resources (e.g., computation and storage) to participant Fairledger and behave honestly for extra profits. Our future work will use game theory to model incentive design and evaluate its effectiveness and robustness.

ACKNOWLEDGEMENT

This work is supported by the U.S. National Science Foundation (NSF) via grant CNS-2039342 and the U.S. Air Force Office of Scientific Research (AFOSR) Dynamic Data and Information Processing Program (DDIP) via grant FA9550-21-1-0229. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U. S. Air Force.

REFERENCES

[1] "Flask: A Pyhon Microframework," http://flask.pocoo.org/.
[2] "gmpy2," https://gmpy2.readthedocs.io/en/latest/index.html.
[3] "pyca/cryptography documentation," https://cryptography.io/.
[4] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2018.
[5] V. Attias, L. Vigneri, and V. Dimitrov, "Implementation study of two verifiable delay functions," *Cryptology ePrint Archive*, 2020.
[6] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Annual international cryptology conference*. Springer, 2018, pp. 757–788.
[7] D. Boneh, B. Bünz, and B. Fisch, "A survey of two verifiable delay functions," *Cryptology ePrint Archive*, 2018.
[8] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.
[9] N. Chen and Y. Chen, "Smart city surveillance at the network edge in the era of iot: opportunities and challenges," *Smart Cities*, pp. 153–176, 2018.
[10] L. Da Xu, Y. Lu, and L. Li, "Embedding blockchain technology into iot for security: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 452–10 473, 2021.
[11] S. Deb, S. Kannan, and D. Tse, "Posat: proof-of-work availability and unpredictability, without the work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 104–128.
[12] A. Fitwi, Y. Chen, S. Zhu, E. Blasch, and G. Chen, "Privacy-preserving surveillance as an edge service based on lightweight video protection schemes using face de-identification and window masking," *Electronics*, vol. 10, no. 3, p. 236, 2021.
[13] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.
[14] R. Han, H. Lin, and J. Yu, "Randchain: A scalable and fair decentralised randomness beacon," *Cryptology ePrint Archive*, 2020.
[15] T. Hewa, G. Gür, A. Kalla, M. Ylianttila, A. Bracken, and M. Liyanage, "The role of blockchain in 6g: Challenges, opportunities and research directions," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*. IEEE, 2020, pp. 1–5.
[16] R. Khalil and N. Dulay, "Short paper: Posh proof of staked hardware consensus," *Cryptology ePrint Archive*, 2020.
[17] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual international cryptology conference*. Springer, 2017, pp. 357–388.
[18] A. K. Lenstra and B. Wesolowski, "A random zoo: sloth, unicorn, and trx," *Cryptology ePrint Archive*, 2015.
[19] J. Long, "Nakamoto consensus with verifiable delay puzzle," *arXiv preprint arXiv:1908.06394*, 2019.
[20] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.
[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.
[22] K. Pietrzak, "Simple verifiable delay functions," in *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
[23] M. Raikwar and D. Gligoroski, "R3v: Robust round robin vdf-based consensus," in *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2021, pp. 81–88.
[24] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996.
[25] M. Stadler, "Publicly verifiable secret sharing," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1996, pp. 190–199.
[26] B. Wesolowski, "Efficient verifiable delay functions," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 379–407.
[27] R. Xu, Y. Chen, and E. Blasch, "Microchain: A light hierarchical consensus protocol for iot systems," in *Blockchain Applications in IoT Ecosystem*. Springer, 2021, pp. 129–149.
[28] R. Xu, S. Y. Nikouei, Y. Chen, E. Blasch, and A. Aved, "Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 564–571.
[29] R. Xu, S. Y. Nikouei, D. Nagothu, A. Fitwi, and Y. Chen, "Blendsps: A blockchain-enabled decentralized smart public safety system," *Smart Cities*, vol. 3, no. 3, pp. 928–951, 2020.
[30] M. Zhou, X. Lin, A. Liu, and Y. Che, "An improved blockchain consensus protocol with distributed verifiable delay function," in *2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI)*. IEEE, 2021, pp. 330–337.