

# Identity Aging: Efficient Blockchain Consensus

Mansoor Ahmed  
University of Cambridge

Kari Kostianen  
ETH Zurich

## ABSTRACT

Decentralized currencies and similar blockchain applications require consensus. Bitcoin achieves eventual consensus in a fully-decentralized setting, but provides very low throughput and high latency with excessive energy consumption. In this paper, we propose identity aging as a novel and more efficient consensus approach. Our main idea is to establish reliable, long-term identities and choose the oldest identity as the miner on each round. Based on this approach, we design two blockchain systems. Our first system, SCIFER, leverages Intel’s SGX attestation for identity bootstrapping in a partially-decentralized setting, where blockchain is permissionless, but we trust Intel for attestation. Our second system, DIFER, creates new identities through a novel mining mechanism and provides consensus in a fully-decentralized setting, similar to Bitcoin. One of the main benefits of identity aging is that it does not require constant computation. Our analysis and experiments show that identity aging provides significant performance improvements over Bitcoin with strong security guarantees.

## 1 INTRODUCTION

Since the popularity of Bitcoin [39], there has been significant interest in development of decentralized digital currencies. Any decentralized currency requires a *consensus mechanism* to prevent double spending. The Nakamoto consensus used in Bitcoin leverages Proof-of-Work (PoW) puzzles [19] and economic incentives: all transactions are recorded to a public ledger organized as a chain of blocks, participants are incentivized to mine (solve puzzles) on the longest branch using rewards, and consensus is eventually reached on one branch, assuming that the adversary does not control majority of the computing power and messages are delivered sufficiently reliably [21]. Experience from the last ten years has shown that Bitcoin works in practice and it is indeed possible to realize a digital currency without a trusted authority.

However, Bitcoin has also serious limitations. When run on a global peer-to-peer network, it’s consensus mechanism requires infrequent and small blocks. As a result, Bitcoin can process only 7 transactions per second. Because consensus is eventual, transaction confirmation takes up to 60 minutes. Such performance limitations make Bitcoin impractical for many payments. The combined Proof-of-Work computation of all miners consumes huge amount of energy — at the time of writing, comparable to a mid-size developed country [17]. Bitcoin has also other problems, such as limited privacy [8, 36], reward centralization [22] and node eclipsing [23].

**Blockchain efficiency.** Consequently, researchers and practitioners have examined numerous alternative blockchain schemes. We focus on *consensus efficiency*, and consider other improvements, like increased privacy [35, 45], complementary to our work.

Blockchain performance can be improved in multiple ways. One common approach is sharding [5, 15, 30, 31, 34, 43], where system participants are divided into committees that process distinct sets

of transaction and a final committee combines the results. The main drawback of PoW-based sharding is that such schemes still require constant computation to establish identities for committee selection. Another common approach is Proof of Stake (PoS) [27, 37], where a consensus leader (or miner) is chosen randomly with a probability that is proportional to owned coins. PoS schemes do not require constant computation, but have their own shortcomings. The leader selection is typically based on an expensive multi-party computation protocol and many PoS systems are vulnerable to new adversarial strategies, such as *nothing-at-stake* attack, where the adversary leverages the fact that extending multiple branches is inexpensive, or *grinding* attack, where the adversary creates blocks or identities in a way that gives him an advantage in leader selection [7]. Many schemes also assume (weakly) synchronous communication — an assumption that does not always hold in global peer-to-peer networks.

Researchers have also explored if trusted computing environments (TEEs), such as Intel SGX [3], can enable more efficient blockchains in a *partially-decentralized* setting. Intel SGX enables protected applications, called *enclaves*, that are isolated from malicious privileged software. In Proof of Elapsed Time (PoET) [1] participants are enclaves that wait a randomly chosen time and the enclave that finishes first becomes the miner. PoET requires no constant computation, but is vulnerable to compromise of one or few SGX processors. In Resource Efficient Mining (REM) [49] each enclave performs some useful processing and produces a matching evidence. REM is more tolerant to CPU compromise, as the reported processing can be capped, but it does not save any energy.

**Our solution.** In this paper, we propose a novel blockchain consensus approach that we call *identity aging*. Our solution is designed to improve blockchain efficiency, fairness and attack-resilience under realistic communication model. Our main idea is simple but powerful: we establish reliable (i.e., Sybil-attack resistant) *identities* that are recorded to a public ledger, the *age* of each identity corresponds to the number of rounds passed since the last successful mining, and on each round one of the oldest identities is chosen to mine the next block. The miner selection is an interactive protocol between a set of eligible miner candidates and a randomly chosen validator committee. The created blocks record active committee participation and thus allow exclusion of inactive identities from miner and validator selection.

On a high level, our idea can be seen as a variation of Proof of Stake, where the mining power of each participant is proportional to the number of controlled identities (instead of coins). In contrast to previous proposals that leverage the notion of aging (e.g., Peercoin [28]), we decouple age from coins to ensure that all coins have the same market value (a necessary property for any currency).

Based on this idea, we design two systems. Our first system, SCIFER (*Semi-Centralized Incentivized Fair Efficient Resilient Consensus*), is designed for a *partially-decentralized* setting, where the blockchain is permissionless, but we trust Intel to run the SGX

attestation service correctly (similar to [1, 49]). We bootstrap reliable identities from the existing infrastructure of SGX processors. The number of controlled identities corresponds to the number of owned SGX processors by each participant, but importantly the processors do not have to perform constant computation. In contrast to previous SGX-based consensus mechanisms [1, 49], our system saves energy and tolerates compromised CPUs. Even platform vulnerabilities like Spectre [12, 29] or Meltdown [33] do not break our system, because the system participants gain no advantage by attacking their own platforms. Later in the paper, we discuss how identities could be bootstrapped also from other infrastructures, like credit cards or mobile subscriptions.

Our second system, DIFER (*Decentralized Incentivized Fair Efficient Resilient Consensus*), is designed for a *fully-decentralized* setting, similar to Bitcoin and many other recent blockchains. Here, we leverage another simple but effective idea: each successful mining event creates an identity reward and such rewards enable enrollment of new identities. This approach allows controlled creation of new identities starting from an initial fair distribution that can be established using an initial PoW phase, for example.

Our analysis shows that identity aging provides similar or better security guarantees than Bitcoin. Short-lived forks may exist, but after a small number of rounds (two or three) the system reaches consensus on one branch with very high probability. We also show that identity aging is resilient against common attacks, such as block withholding, nothing-at-stake, grinding, long-range attack. Such security properties are provided under an realistic communication model, where delivery of all messages cannot be guaranteed. Moreover, identity-aging removes incentives for pooled mining which improves decentralization.

The performance of our consensus mechanism depends primarily on the underlying network and the size of the system. We measured message delivery using a globally-distributed peer-to-peer that we built and estimated throughput and latency in identity aging. For deployments of 10,000 active participants (the approximate size of the current Bitcoin network), transaction confirmation takes 10-15 seconds and our system can process roughly 1500 transactions per second. In comparison to Bitcoin, both latency and throughput are improved two orders of magnitude.

Our approach has also minor drawbacks like small privacy loss due to correlation of mining events (but not transaction events), increased susceptibility to DoS attacks due to more predictable miner selection, and reduced performance due to the required tracking of active identities for the largest deployments.

**Contributions.** To summarize, our paper makes the following main contributions:

- *New consensus approach.* We propose identity aging as a novel blockchain consensus mechanism.
- *Partially-decentralized blockchain.* We design a new blockchain system, called SCIFER, by bootstrapping identities from the Intel SGX attestation infrastructure.
- *Fully-decentralized blockchain.* We design another new blockchain system, called DIFER, by leveraging an initial fair distribution of identities and identity mining for fully-decentralized deployments.

- *Analysis and evaluation.* We show that identity aging provides significant performance improvements and strong security guarantees assuming a realistic communication model.

The rest of the paper is organized as follows. Section 2 provides an overview of identity aging. Sections 3 and 4 describe our two systems. Section 5 provides security analysis and Section 6 performance evaluation. Section 7 includes discussion, Section 8 reviews related work, and Section 9 concludes the paper.

## 2 IDENTITY AGING

This section provides an overview of our approach. We explain our goals, the adversary and communication models, and the main idea of consensus using identity aging.

### 2.1 Goals

Our goal is to design a *consensus mechanism* for a decentralized blockchain. The blockchain can be used for a digital currency or any other application that enables significant rewards that incentivize behavior similar to monetary rewards. We assume that the blockchain is used to record transactions, but our consensus mechanism is agnostic to their type that can be pseudonymous as in Bitcoin or anonymous as in recent proposals like [26, 35].

We consider two separate trust models. The first is a *partially-decentralized* setting, where the blockchain consensus is maintained by a permissionless set of system participants in a decentralized manner, but we rely on Intel to manufacture processors and run the SGX attestation service correctly. In this setting, our main goal is to improve the efficiency and security of previous SGX consensus mechanisms [1, 49]. In particular, we want to design a solution that tolerates compromised CPUs and requires no constant computation.

Our second trust model is a *fully-decentralized* blockchain, similar to Bitcoin and many other recent blockchain proposals. In this setting, we have three main goal. The first goal is to eliminate the need for constant computation that is a major problems in all PoW systems, such as Bitcoin [39], Bitcoin NG [20] and various sharding schemes [5, 15, 30, 31, 34, 43]. The second goal is to provide a scheme that is robust to known attack strategies, such as nothing-at-stake, grinding, and long-range attack, that especially affect many of Proof-of-Stake systems [7]. The third goal is to design a system that works in global peer-to-peer networks where reliable communication cannot be always guaranteed.

### 2.2 Adversary and Communication Model

We consider economically-driven users, who seek to maximize their own gain and whose investment in breaking the system is proportional to the gained value — or the potentially the lost value in case of being caught from cheating. We model system participants as *processors* and consider an adversary that controls a significant fraction  $\alpha$  of all  $n$  processors (e.g.,  $\alpha = 0.33$ ). We denote the number of adversary-controller processors  $m = \alpha \cdot n$ . A single malicious user may have purchased  $m$  processors or a number of users may collude and use their combined  $m$  processors together. In the latter case, we consider the set of colluding users as the adversary.

We assume an adversary that can add new processors that he controls to the system, as long as his fraction of all processors stays within  $\alpha$ . The adversary *cannot* choose for every time window  $t$

which  $m$  processors he controls (e.g.,  $t = 5$  seconds). We note that this is in contrast to some of the recent schemes that consider a *fully-adaptive* adversary [27, 37] that can freely choose controlled participants for each time window. Our take is that a such fully-adaptive adversary is interesting, and worth studying, but often not realistic. In practice platform compromise is hard to detect and repair. Furthermore, a compromise of one computing platform does not mean that another is no longer in control of the adversary.

In contrast to the previous SGX consensus schemes [1, 49], we assume that the adversary can break SGX protections on all  $m$  processors that are under his physical possession. For example, through hardware attacks or discovered platform vulnerabilities [12, 29, 33] the adversary can extract secret keys from CPUs and modify enclave control flow. We consider such attacks time consuming and expensive, but realistic in applications like digital currencies, where the economic payoff can outweigh the cost of the attack. We stress that SGX was designed to protect enclaves from malicious privileged software, but not from physical tampering [13].

We assume that the participants communicate over a global peer-to-peer network, similar to Bitcoin. Within each time window  $t$  each processors is able to communication with all other processors except a small fraction  $\beta$  (e.g.,  $\beta = 0.05$ ). For example, in the (arguably slow) Bitcoin network broadcasted blocks are received by 95% of the nodes after 40 seconds [16]. Finally, we assume that participants have loosely synchronized clocks.

## 2.3 Main Idea

Our consensus mechanism is based on long-term, reliable identities. By the term “reliable” we mean identities that the adversary cannot create without restrictions (*Sybil attack* [18]) and at the same time the system should be permissionless, allowing any user to participate in blockchain consensus. More precisely, identity establishment should be such that an adversary that controls fraction  $\alpha$  of all processors should be able to control at most  $\alpha$  of all identities.

We describe two ways to establish such reliable identities. The first is bootstrapping from an existing infrastructure of Intel SGX attestation (Section 3) and the second is mining the identities starting from an initial fair distribution of identities (Section 4). We represent identities as public keys and every identity creation event is recorded, in a verifiable manner, to the public ledger (blockchain). For each identity an *age* can be determined by counting the number of blocks since its creation or the last successful block mining.

Figure 1 illustrates the high-level operation in identity aging. After the initialization phase (that guarantees an initial fair distribution of identities), the system proceeds in rounds of fixed duration  $t_r$  (e.g.,  $t_r = 5$  seconds). In the beginning of each round, every identity checks if it is a *miner candidate*. As candidates we select a set of oldest identities with recent recorded activity in the ledger. On the beginning of each round, we also select a random sampling of *validators*.

Each miner candidate broadcasts an Intent message that is bound to one chain branch and set of proposed transactions. Based on the received intent messages, the validators pick the oldest candidate and send back to it a Confirm message. If a miner candidate receives the required quorum  $q$  of confirmation messages, it is allowed to create and broadcast a new Block. The miner includes to the

block new transactions and the received confirmation messages. By parsing the chain, every participant can verify which identities have recently recorded activity in the form of confirmations. To ensure liveness, we exclude inactive participants from miner candidate and validator selection.

New participants can join the system at any time by sending an Enroll message that contains a publicly verifiable proof of a new identity (SGX attestation or mined token). Miners include such new identities to created blocks (eventually).

## 3 SCIFER SYSTEM

In this section we describe our first system, called SCIFER, that is designed for a partially-decentralized setting. For bootstrapping reliable identities we leverage Intel’s SGX attestation infrastructure. Later in the paper (see Section 7) we discuss how similar identities could be bootstrapped also from other infrastructures, such as credit cards or mobile subscriptions.

### 3.1 Bootstrapping from SGX

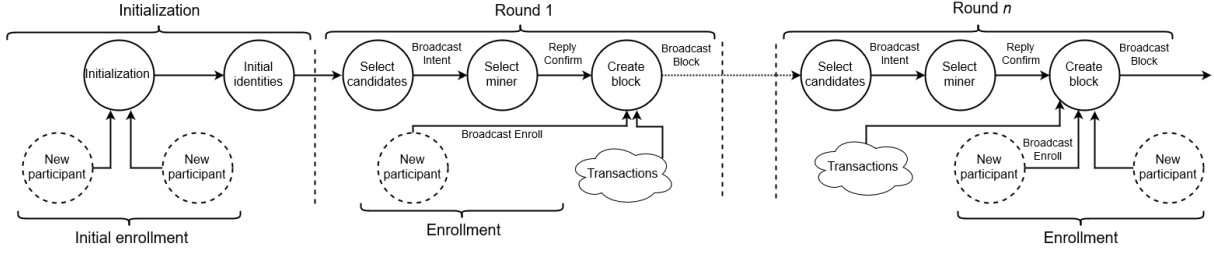
Intel’s Software Guard Extensions (SGX) [3, 13] enables execution of protected applications, called *enclaves*, in isolation from any untrusted system software such as a malicious OS. SGX protects integrity of enclave execution and confidentiality of enclave data. Protections in the processor prevent a malicious OS from directly reading or modifying enclave memory at runtime. Processors have also keys that allow encryption for persistent storage (sealing).

For our solution, the most relevant part of SGX is its *remote attestation* mechanism. During manufacturing, each SGX processors is equipped with a key that is certified by Intel. Attestation is a protocol where a remote entity can verify that specific enclave code runs on a genuine SGX processor. In attestation, the processors signs a statement over the enclave’s software configuration, which was recorded during its initialization, using the certified processor-specific key. The verifier forwards the signed statement to Intel Attestation Service (IAS), an online service run by Intel, that sends back signed attestation evidence, assuming the remote attestation was successful.

SGX attestation uses group signatures and it is anonymous, in the sense that it does not identify the attested hardware platform (e.g., the processor serial number or similar identifier) [25]. Importantly for our solution, SGX attestation supports also a *linkable* mode of attestation that allows the remote verifier to test if the currently attested processor has been previously attested without identifying it. We provide further SGX details in Appendix B.

In SCIFER, we leverage the linkable mode of SGX attestation for bootstrapping identities. As identities we use public keys of key pairs that are generated inside enclaves. We bind these keys to the attestation protocol and save the attestation evidence (signed by IAS) to the blockchain. Given such evidence, anyone can verify that the same processor is enrolled at most once.

Interestingly, our solution does *not* require enclave data confidentiality or execution integrity. We use sealing to protect the IAS access credential, but its secrecy is not relevant for blockchain consensus. Thus, our system tolerates adversaries that can compromise their own processors. Next, we describe the SCIFER system in detail.



**Figure 1: Identity aging overview.** The initialization phase guarantees an initial and fair distribution of identities that are recorded to the ledger. After that, the system proceeds in rounds of fixed duration. During each round, one of the oldest identities is selected as the miner that creates and broadcast a new block. New participants can join the system on any round.

### 3.2 Initialization

A new blockchain instance can be initialized by any (potentially untrusted) entity that we call chain creator. The creator registers with Intel and obtains an access credential  $c_a$  for the Intel Attestation Service (IAS). At registration, the creator specifies that linkable mode of attestation is used.

The creator chooses  $n_0$  platforms as the initial system members. These platforms install enclave code that creates an asymmetric key pair, seals the private key  $sk_i$ , and exports the public key  $pk_i$ . The creator performs a remote attestation on each of the selected platforms. During attestation, each platform supplies a hash of  $pk_i$  as the USERDATA to be included as part of the QUOTE structure  $Q_i$  (see Appendix B). If the attestation is successful, IAS signs  $Q_i$  that includes a pseudonym  $p_i$  for the attested platform. The attested enclaves send their public keys  $pk_i$  to the creator.

The creator checks if the public keys match the respective hashes reported in each QUOTE structure  $Q_i$  and if all attested platforms are separate, i.e., each  $Q_i$  has a different pseudonym  $p_i$ . If this holds, it constructs a genesis block  $\text{Block}_0 = (pk_1, Q_1, pk_2, Q_2, \dots, h_e, id)$  that includes public keys  $pk_i$  and the signed quote structures  $Q_i$  for each initial member, a hash of the enclave code  $h_e$ , and a hash  $id$  over all elements that serves as the chain identifier. The creator publishes the genesis block and sends the IAS access credential  $c_a$  to the attested enclaves that seal it.

### 3.3 Enrollment

New participants can join the system on any round. The joining platform installs the enclave code defined by  $h_e$ , creates a key pair, seals the private part  $sk_n$ , exports the public part  $pk_n$  and contacts one of the current members (e.g., by broadcasting to the peer-to-peer network). The current member performs remote attestation on the new platform using  $h_e$  as the reference. During attestation, the enclave of the new platform supplies a hash  $h(pk_n || r || h_b)$  as its USERDATA, where  $r$  and  $h_b$  are the round number and hash of the latest block (to bind the enrollment to a specific branch). If the attestation is successful, the existing member obtains a signed QUOTE structure  $Q_n$  from IAS, including an attestation pseudonym  $p_n$ . It verifies that the pseudonym  $p_n$  does not appear in any of the previously enrolled platforms in the chain (recall that each  $Q_i$  is saved to the ledger). The verifier sends  $c_a$  to the attested enclave and constructs an enrollment message  $\text{Enroll}_n = (Q_n, pk_n, r, h_b)$  that serves as a membership proof and broadcasts this to the network.

Once the membership proof is included to a new mined block, a new identity is established.

### 3.4 Mining Protocol

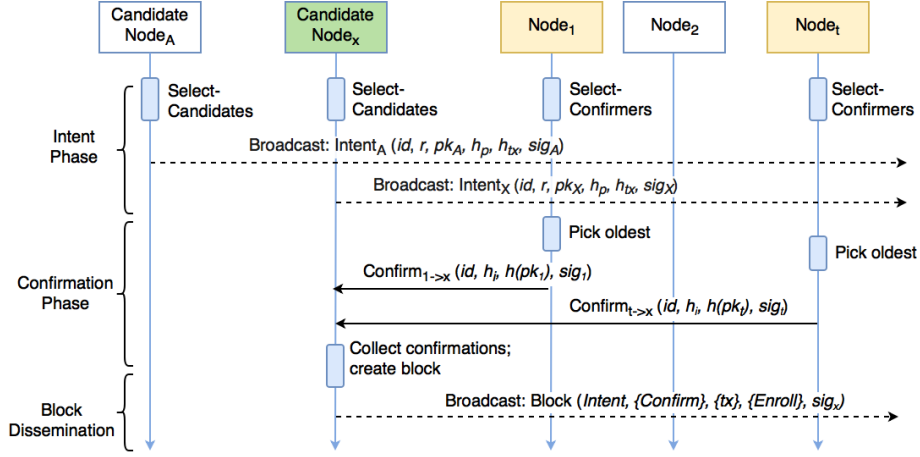
After initialization, the system proceeds in rounds of fixed length  $t_r$ . Figure 2 illustrates the mining protocol that consists of three phases. In the beginning of the round, every identity tests if it is a miner candidate or validator for that round using `SelectCandidates` and `SelectValidators` algorithms that are described below. The number of miner candidates  $C$  can be a small fixed value (e.g.,  $C = 10$ ), while the size of the validator committee  $V$  depends on the number of currently active identities  $n_a$  (e.g.,  $V = 0.01 \times n_a$ ). We explain the details of the miner candidate and validator selection below.

As in most permissionless blockchains, every successfully mined block (that will become part of the longest branch) will be rewarded. The type and amount of the reward depends on the blockchain application. For example, in a digital currency the reward could be a significant amount of new coins.

**Intent phase.** The first phase of each round is an intent phase, where each miner candidate  $c$  broadcasts an intent message  $\text{Intent} = (id, pk_c, r, h_p, h_{tx}, sig_c)$  that contains the chain identifier  $id$ , the candidate's identity  $pk_c$ , the current round number  $r$ , the branch of the chain that the candidate proposes to extend indicated by the hash of the previous block  $h_p$ , hash of the transactions  $h_{tx}$  the miner proposes to include to the mined block, and the candidate's signature  $sig_c$  over these elements. In case multiple chain branches exist, the candidate uses the `SelectBranch` algorithm, described below, to choose which branch to extend.

**Confirmation phase.** Each validator  $v$  verifies all Intent messages received during the intent phase by checking that the sender is a valid miner candidate for this round. Among the received intent messages, the validator selects the candidate with the highest age (i.e., the identity with most rounds from previous mining or enrollment), and sends to it a confirmation message  $\text{Confirm}_{v \rightarrow c} = (id, h_i, h(pk_v), sig_v)$  that indicates that validator  $v$  has confirmed candidate  $c$ . The message contains the chain identifier  $id$ , hash of the intent message  $h_i$ , the validator identity  $h(pk_v)$ , and a signature  $sig_v$  over the previous elements.

In case multiple candidates have the same age, the validator picks a pseudorandom number using the *stable* part of the chain as a seed, sorts the candidates of same age based on the binaries of their (public key) identities, and selects one of them based on the



**Figure 2: Mining protocol.** Each miner candidate broadcasts Intent messages and validators reply with Confirm messages. A node that receives the required quorum of confirmation becomes a miner that can create and broadcast a new block.

pseudorandom number. The purpose of this process is to ensure that all (non-malicious) validators choose the same candidate. The seed for pseudo-randomness can be computed as  $h(\text{Block}_{r-d})$  where  $r$  is the current round number and  $d$  denotes the maximum depth of forks (see Section 5 for analysis). If one validator received intent messages that refer to more than one chain branches, the validator picks the branch (and the intent) to confirm using the SelectBranch algorithm.

**Block dissemination phase.** The final phase of the round is block creation and dissemination. If a candidate  $c$  received the required quorum of  $q$  confirmations, it becomes an eligible miner for the current round. The miner creates a new block  $\text{Block}_r = (\text{Intent}, \{\text{Confirm}\}, \{tx\}, \{\text{Enroll}\}, sig_c)$  that contains the intent message, received confirmation messages, new transactions  $tx$ , membership proofs of new enrolled platforms, and a signature  $sig_c$  over the previous elements by the miner. The miner broadcasts the created block.

### 3.5 Mining Algorithms

**Candidate selection.** The SelectCandidates algorithm parses the blockchain based on two adjustable parameters: activity threshold  $T_a$  and the number of candidates  $C$ . The activity threshold can be, e.g.,  $T_a = 3000$  rounds that would match a few hours of system operation.

First, SelectCandidates selects the chain branch to use based on SelectBranch (described below). Then, it parses the selected branch starting from the newest block. For the  $T_a$  latest blocks, the algorithm adds senders of confirmation messages to a list that represents members with recorded activity. Next, the algorithm parses the chain starting from the oldest block. When block mining or enrollment by any of the identities in the list is encountered, the identity's age (i.e., the number of blocks from the latest block) is saved to the list. This process is continued until every identity in the list is assigned an age. Finally, the list is sorted by age and  $C$  oldest identities, and their ages, are returned.

**Validator selection.** The SelectValidators algorithm selects a committee of  $V$  validators (e.g.,  $V = 100$  for a system of 10,000 active identities). SelectValidators computes a list of recently active identities as explained above. After that, it selects  $V$  identities using standard simple random sampling (with replacement), where all listed members are sorted based on their public key binary and the required random number is generated using the *stable* part of the blockchain as the seed. The seed for pseudorandomness is computed as  $h(\text{Block}_{r-d})$ .

**Branch selection.** The SelectBranch algorithm first verifies the correctness of each branch using VerifyBranch algorithm described below. Then, it computes a length for each of the branches which is defined by the number of rounds with missing blocks. The longest branch is always selected. If more than one branch has the same length (i.e., an equal number of missing blocks), the algorithm chooses the branch using a pseudorandom number whose seed is derived from the stable part of the chain.

**Branch verification.** The VerifyBranch algorithm checks that a given chain branch is correctly constructed. As the first step it traverses the chain and checks that each block contains a correct hash of the previous block. The next step is to verify that all enrolled platforms have valid membership proofs (correct attestation evidence and unique attestation pseudonym in the case of SGX identities). After that, the algorithm verifies that the miner of each block was an eligible candidate on that round (SelectCandidates), the block contains the required  $q$  confirmation tokens, the confirmation token contain the hash of the intent message included to the block, and the validators that were eligible confirmers on that round (SelectValidators).

### 3.6 Re-enrollment

If an enrolled identity does not participate in the system by sending confirmation messages during  $T_a$  rounds (e.g., few hours), it will be excluded from miner and validator selection. In such cases, the platform can perform the enrollment protocol again. In re-enrollment, a chosen verifier checks that the IAS service returns

the same pseudonym  $p_i$  that was used for this identity (public key  $pk_i$ ) during enrollment. If this is the case, the verifier can create and broadcast a new membership proof with a flag that indicates re-enrollment. Once such re-enrollment is recorded to a block, the platform is included to miner candidate and validator selection again.

## 4 DIFER SYSTEM

In this section, we design a second system, called DIFER, that is based on the same idea of identity aging but tailored for the fully-decentralized setting. The key problem that we need to solve is how to establish the needed reliable identities without bootstrapping from an existing infrastructure such as SGX attestation. Identity creation should be permissionless in the sense that any user can enroll, but controlled in the sense that the adversary that controls a fraction  $\alpha$  of processors can create at most  $\alpha$  of the new identities.

### 4.1 Mining Identities

Our approach is to *mine* identities, that is, reward successful block creation events with new identities, assuming an initial fair distribution of identities, where the adversary controls at most  $\alpha$  of the initial identities. Such initial distribution can be established using a Proof-of-Work phase.

A straw-man solution is to apply the idea of Proof of Stake such that successful mining is rewarded with new coins and each owned coin directly corresponds to one identity in the system. As creation of new coins is recorded to the ledger, on each round the owner of the oldest coin can be chosen as the miner, following the principle of identity aging. Such approach would be resilient against Sybil attacks, as mined new coins and identities are proportional to the initial and fair distribution of identities.

However, this straw-man solution has two major limitations. The first is that, to support identity-aging miner selection, the coins must be made indivisible units. Indivisible coins increase transaction size and the ledger processing overhead and reduce system performance. In a system with total value, say, one billion US dollars, and the smallest currency unit matching to one US cent, a hundred billion coins and identities would be needed.

The second drawback is that different coins of same denomination would have different market values. For example, if a coin is old and soon eligible for mining, its market value is arguably higher than that of a new coin. Fungibility (i.e., that the units of the same denomination all have the same value and are interchangeable) is an important property of any monetary system.

To avoid the above problems, we adopt an approach, where the creation of new identities is proportional to the currently controlled identities, but we *decouple* the direct linkage between the mined coins and the identities, to enable coin divisibility and fungibility. Every mining operation creates new coins (or similar rewards depending on the blockchain application), and additionally an *identity reward*. Sufficiently many identity rewards ( $\geq 1$ ), in turn, can be used to enroll a new identity to the system.

The tokens can be used in two ways. First, the miner (the owner of tokens) can enroll a new identity for himself. Second, owner of the tokens can sell them to a new user that wants to participate in the system, and the new user can then enroll a new identity into

the system. As the tokens that create new identities are decoupled from coins, coins can be divided for transaction efficiency and coins are also interchangeable.

Next, we explain our DIFER system in more detail. The mining protocol and algorithms are the same as before (Section 3) with only minor modifications. Initialization and enrollment are specific to the DIFER system.

### 4.2 Initialization

The initial distribution of identities can be established using a preliminary Proof-of-Work (PoW) mining phase. Every successful PoW mining creates a block  $\text{InitBlock}_i = (h_{i-1}, pk_i, pow)$  that contains a hash  $h$  of the previous block, public key  $pk_i$  of the miner that defines a new identity in the system, and a Proof-of-Work solution  $pow$ . Initial blocks do not contain transactions, they are broadcasted to the peer-to-peer network, and their recipients continue mining based on them. Such blocks have an order in the chain and thus every initially mined identity has an associated age. The initial distribution of identities is the series of public keys  $pk_0, pk_1, \dots, pk_{n_0}$  from the sequence of blocks  $\text{InitBlock}_0 = (pk_0, pow_0)$ ,  $\text{InitBlock}_1 = (h_0, pk_1, pow_1)$ ,  $\dots$ ,  $\text{InitBlock}_{n_0} = (h_{n_0-1}, pk_{n_0}, pow_{n_0})$ .

The initialization phase can last as long as sufficiently many ( $n_0$ ) identities have been created. The difficulty of the computational puzzle can be set such that frequency of mining events is low to make forks during initialization rare. Potential forks can be solved as in Bitcoin. If further incentives, besides the initial identities, for the initial mining work are required, every successful mining operation can be additionally rewarded with coins or similar application-specific rewards.

### 4.3 Mining and Enrollment

After the initialization phase, the system proceeds in mining rounds as detailed in Section 4.3. The only difference is that each block creation is rewarded with the normal mining rewards (such as new coins), but also with a new identity token. By adjusting the number of required tokens it is possible to control the rate of new identities entering the system.

We design the identity rewards and enrollment as follows. Once a participant  $pk_m$  has mined the required number of blocks with the same identity, it creates a new key pair  $pk_n, sk_n$  that it uses for the enrollment. The enrollment message  $\text{Enroll} = (h_1, h_2, \dots, h_p, pk_n, sig_m)$  contains a set of hashes  $\{h_i\}$  that refer to the previously mined blocks by identity  $pk_m$  (i.e., blocks with unused identity rewards), the public key of the new identity  $pk_n$ , and a signature  $sig_m$  over these elements using the private key of the identity  $pk_m$  that mined the previous blocks. The participant broadcasts the enrollment message, and once it is included to a new block, a new entity exists in the system.

The same mechanism can also be used to allow new participants to join the system. The owner of the identity tokens can sell them to another participant by including a public key received from the buyer to the enrollment message. The payment from the buyer to the seller can be realized, e.g., using fiat money. The buyer of the tokens should give the money to the seller only once he sees the correct enrollment message in the chain in a block that has been extended with  $d$  valid blocks to prevent double selling of tokens.

Chain verification works as already explained in Section 3 with the following additional checks for the correctness of enrollment. Validity of the enrollment message requires that (i) the set of hashes  $h_i$  refer to previous valid blocks, (ii) the previous valid blocks have not been used to create a new identity already, (iii) all the referred previous blocks have been created by the same identity  $pk_m$ , and (iv) the signature of the enrollment message is correct.

## 5 SECURITY ANALYSIS

In this section analyze the security of identity aging based mining. First, we show that identity aging provides consensus assuming a fair distribution of identities. Then, we discuss known attack strategies. And finally, we show that identity creation in SCFIER and DIFER provide reliable identities.

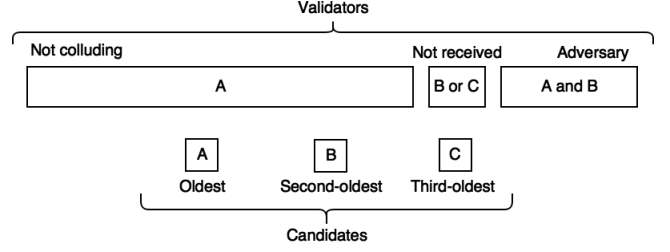
### 5.1 Stability

For the analysis of identity aging we use the definition of *stability* from Bonneau et al. [10] with minor adaptation. A consensus mechanism is called stable, if it provides:

- *Eventual consensus.* At any time, all honest nodes agree upon a *prefix* of what will eventually become the valid blockchain. Due to the possibility of temporary forks, at any given moment we do not require that the entire current chain is the prefix.
- *Exponential convergence.* The probability of a fork at depth  $d$  in the chain is  $O(2^{-d})$ . This gives participants high confidence that after a transaction is added to a block and the block is extended with a small number of valid blocks, the transaction is permanently part of the chain.
- *Correctness.* All the blocks in the prefix of the eventually valid chain will only include valid transactions.
- *Liveness.* New blocks continue to be added and valid transactions will be included in the blockchain within a reasonable amount of time.
- *Fairness.* On expectation, a miner with fraction  $\alpha$  of all processors will mine fraction  $\alpha$  of all blocks, and collect a similar fraction of mining rewards.

**Consensus and convergence.** To explain the possible scenarios in miner confirmation, Figure 3 shows an example where the three oldest miner candidates are A, B and C. The validator committee is sampled based on a pseudorandom value on each round. On the average,  $\alpha$  of the sampled validators are adversary-controlled and  $\beta$  of the validators may not receive Intent sent by the oldest candidate A. The non-colluding validators who received all messages (fraction  $1 - \alpha - \beta$ ), always confirm A as the miner. Those non-colluding validators that did not receive all messages may confirm one another candidate (B or C). The adversary-controlled validators may confirm more than one candidate (A and B), although such equivocation leaves evidence that can be easily used to penalize malicious identities (see Section 7).

The exact number of validators on each of the above discussed groups varies on each round. In a rare case, that we call *adversarial sampling*, enough adversary-controlled identities may be chosen to the validator committee, and thus also the second-oldest candidate B may received the required quorum of confirmations, causing two eligible miners (A and B) on the same round and a fork in the



**Figure 3: Miner confirmation example.** Non-colluding validators always confirm the oldest miner candidate (A). Validators who did not receive all Intent messages may confirm another candidate (B or C). Adversary-controlled validators can confirm multiple candidates (A and B).

chain. We denote the probability of adversarial sampling as  $\Pr(\text{AS})$ . Assuming an unbiased pseudorandomness seed and sufficiently large  $n$ , this probability can be computed as:

$$\Pr(\text{AS}) = \sum_{i=0}^{V-q} \left( \binom{V}{q+i} (\alpha + \beta)^{q+i} (1 - \alpha - \beta)^{V-q-i} \right),$$

where  $V$  is the number of validators,  $q$  is the quorum size,  $\alpha$  is the fraction of colluding identities, and  $\beta$  is the fraction that cannot communicate with the oldest candidate. For example, when  $\alpha = 0.33$ ,  $\beta = 0.05$ ,  $V = 100$  and  $q = 54$ , then  $\Pr(\text{AS}) = 0.0008$ . Such sampling would take place, on the average, every 1200 rounds.

Extending both forked branches requires another similar sampling. The probability of consecutive sampling decreases exponentially and the probability of three consecutive samplings is very low ( $5.78 \times 10^{-10}$ ). For such parameter values we consider the maximum depth of forks  $d = 3$ .

Once the adversary can no longer create a new block within the forked branch, two options exist. First, it may publish his branch. The next miner chooses randomly between two branches of equal length and extends the selected branch (recall *SelectBranch* from Section 3.5). On the following rounds, all honest members extend this longest branch. Second, the adversary may choose to release it later (block withholding). However, as the honest nodes will always extend the longest branch and the adversary can only extend his branch with few blocks, withholding strategy cannot create deeper forks.

Increasing the quorum size  $q$  reduces the possibility of forks further, but weakens liveness as discussed below. Also a larger validator committee size  $V$  reduces the possibility of forks, but causes an increased performance overhead due to more included confirmation messages (see Section 6).

The adversary can enroll multiple new identities to the system within a short period of time and consequently an identity controlled by the adversary can be selected as the miner on several consecutive rounds. In such cases, the probability that the adversary can mine more than one block on each of these rounds is also low ( $\Pr(\text{AS})$ ), because the non-colluding validators confirm only one Intent per round.

**Correctness.** If a miner publishes a block containing invalid transaction, none of the non-colluding nodes will accept this block.

As explained above, the probability that the adversary-controlled nodes can confirm multiple successive invalid blocks decreases exponentially, and thus the stable part of the chain (i.e., the prefix up to  $\text{Block}_{r-d}$ ) contains only valid transactions.

**Liveness.** Successful block generation on each round requires that the oldest miner candidates receives the required quorum of confirmations from the validators. To analyze the probability for this, we first consider the benign case where all validators (also the adversary-controlled identities) confirm the oldest Intent message they receive. We denote the probability that more than  $V - q$  validators will be sampled from those identities that did not receive the intent message as *benign liveness violation*  $\text{Pr}(\text{BL})$  and compute it as:

$$\text{Pr}(\text{BL}) = \sum_{i=0}^q \left( \binom{V}{V-q+i} \beta^{V-q+i} (1-\beta)^{q-i} \right).$$

Given the previous example parameters, this probability is negligible ( $6.96 \times 10^{-33}$ ).

Next, we consider the case where the adversary reduces the probability of successful mining by not sending confirmation messages. Such *adversarial liveness violation* probability  $\text{Pr}(\text{AL})$  can be computed as:

$$\text{Pr}(\text{AL}) = \sum_{i=0}^q \left( \binom{S}{S-q+i} (\alpha + \beta)^{S-q+i} (1 - \alpha - \beta)^{q-i} \right).$$

Given the previous example parameters, the adversary can prevent mining with probability 0.062, that is, on the average every 16th round. The probability of successive samplings that allow the adversary to prevent mining reduces exponentially. The probability to prevent mining on five successive rounds, for example, is  $9.37 \times 10^{-7}$ . If the adversary continues this strategy longer than the activity period  $T_a$ , his identities will be considered inactive and they can no longer reduce the mining probability for other participants. Additionally, participants have an incentive to provide confirmations that ensure eligibility for mining.

The quorum size  $q$  presents a trade-off between fork depths that determine transaction confirmation latency and liveness that describes how effectively the adversary can prevent mining and thus temporarily reduce system throughput, as illustrated in Figure 4. Smaller  $q$  causes higher probability for forks, while larger  $q$  allows the adversary to prevent mining with higher probability. Figure 4a shows an example case with  $V = 100$  validators and  $\alpha = 0.33$ . A quorum size  $q \approx 50$  provides both low probability for forks and low probability that the adversary can prevent mining several rounds. Figure 4b shows that increasing the validator committee size to  $V = 200$  reduces these probabilities further. Figure 4c shows that if the adversary controls a larger fraction of the active identities ( $\alpha = 0.4$ ) these probabilities increase significantly. In each of the figures, the black dotted line highlights a threshold of  $10^{-8}$  that we consider an acceptable probability for forks in many scenarios.

In a rare case, where a significant portion of the network simultaneously goes offline (e.g., a problem with a major ISP), our system will experience a temporary drop in throughput, as the candidate nodes have low probability of receiving the required confirmations and thus on many rounds no miner will be chosen. Once the nodes are back online, the system will regain its normal throughput.

**Fairness.** Identity aging essentially performs a round-robin miner selection, and therefore each participant with a fraction  $\alpha$  of enrolled processors will mine, on expectation, a fraction  $\alpha$  of blocks. The adversary can attempt to violate system fairness in two ways. The first approach is that the adversary does not include Enroll messages from the targeted victim participant to his blocks. This approach can delay enrollment of a new identity by a few rounds, but not prevent it, and thus such approach does not violate fairness in long term.

The second approach is that the adversary does not include confirmation messages from the victim to his blocks and after  $T_a$  rounds the victim is excluded from miner candidate selection and has to re-enroll. Such *adversarial exclusion* probability  $\text{Pr}(\text{AE})$  can be computed as:

$$\text{Pr}(\text{AE}) = \left( 1 - \frac{V}{n} \right)^{T_a(1-\alpha)}.$$

Assuming  $n = 10,000$  active participants (similar to Bitcoin) and our example parameters ( $V = 100$  and  $T_a = 2000$ ), the adversarial exclusion probability  $\text{Pr}(\text{AE}) = 1.41 \times 10^{-6}$ . Active participants will be excluded from mining very rarely.

If the system grows significantly larger than that (e.g.,  $n = 100,000$ ) we either need to increase the activity period or the validator committee size to keep  $\text{Pr}(\text{AE})$  low. The first approach increases blockchain processing time, while the second implies reduced throughput, as more confirmation messages are included to blocks. We analyze these two trade-offs in Section 6.

## 5.2 Attack Discussion

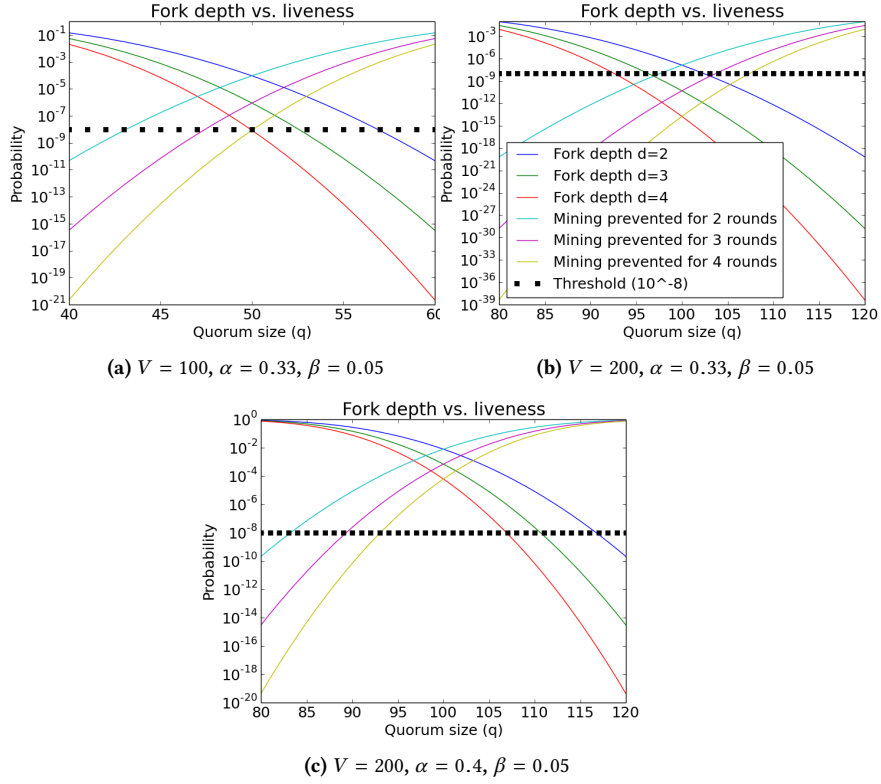
**Nothing-at-stake attack.** In many PoS systems, once a specific participant is chosen as the miner, block creation does not consume significant resources. Consequently, the chosen miner has an incentive to create multiple blocks and extend several branches to maximize his chances of extending the eventually winning branch and collecting the mining reward. Such adversarial behavior is called *nothing-at-stake* attack.

In identity aging, the miner indicates the extended chain branch  $h_p$  in the broadcasted Intent. Non-colluding validators confirm only one Intent per round and our analysis above shows that the probability that the adversary-controlled validators confirm multiple intents is small and due to long-term identities such behavior can be easily penalized (see Section 7). Therefore, the chosen miner can, with high probability, only extend one branch.

**Grinding attack.** A common approach in committee-based consensus systems is to derive the required randomness for committee selection from the stable part of the chain. Some such systems can be vulnerable to a *grinding* attack, where the current miner tries out many different block values and publishes the one that improves his chances of being selected as the miner soon again. Alternative, the participants could try to choose their identities (e.g., public keys) such that they gain an advantage in miner selection.

In identity aging, the candidate indicates the set of transactions  $h_{tx}$  that he proposes to include to the mined block. This serves as a commitment, that the miner candidate cannot change after Intent broadcast. The validators include a random value to their confirmation tokens that must be included to the created block. Thus, the final block value cannot be chosen by the miner candidate





**Figure 4: The quorum size  $q$  provides a trade-off between the probability of forks and liveness. Increased  $V$  enables both shallow forks and good liveness. Larger  $\alpha$  means that only one of the above is possible.**

alone and similar grinding is not possible. Also identities must be chosen at the time of enrollment which is prior to committee selection.

**Long-range attack.** In some schemes identities may become practically worthless to their owners over time. For example, if an identity is used for mining, it has collected reward and exchanged it to goods, the identity has no longer value for its owner. An adversary might be able to purchase a large fraction (more than  $\alpha$ ) of such identities and use them to re-write mining events from the past. In identity aging, identities continue to have value and thus similar long-range attacks are unlikely.

### 5.3 Identity Creation

**SCIFER.** The adversary may attempt to enroll non-SGX platforms to a chain. Such false enrollment would fail, as the IAS service will not return a signed QUOTE needed for enrollment. Enrolling the same SGX platform multiples times would fail as well, because the IAS service would return the same pseudonym  $p_n$  that is already recorded for another identity in the chain. The third alternative is to enroll the same SGX platform to multiple chains and try to reuse enrollment from one chain to another. Because the QUOTE contains the chain identifier  $id$ , this approach would also fail. We conclude that the adversary cannot enroll more identities than the number of SGX platforms he controls.

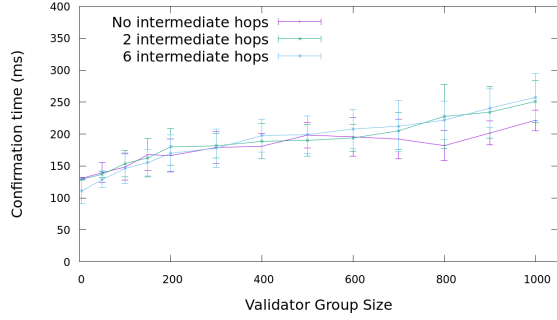
Perhaps a surprising property of our system is that the adversary does not gain advantage by breaking into his own SGX processors. Besides attestation, we only use enclaves for the protection of the IAS credential. Leakage of this credential does not allow the adversary to create additional identities. A malicious chain creator could initialize an invalid chain, where all members are not SGX processors. However, any legitimate participant can detect this due to missing QUOTES in the genesis block and neglect the chain.

The partially-decentralized SCIFER system has one centralized component: Intel's attestation service (IAS). If this service is unavailable temporarily, new nodes cannot be enrolled during its downtime, but the system can mine new blocks and thus process incoming transactions.

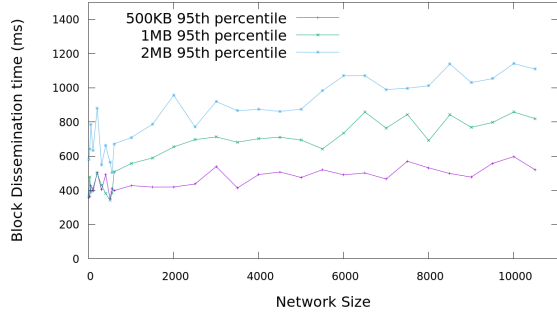
**DIFER.** Identity mining in DIFER requires an initial fair distribution of identities. This can be achieved using standard Proof-of-Work phase. For example, with a computational puzzle that takes on the average 30 minutes, an initialization phase that creates  $n_0 = 1000$  identities would last approximately 21 days. Our above analysis shows that identity aging provides fairness, and thus the distribution of identity tokens and new enrolled identities is also fair.

## 6 PERFORMANCE EVALUATION

In this section we evaluate the performance of identity aging. First, we measure message delivery times using a globally-distributed



(a) Intent and confirmation time.



(b) Block dissemination time.

**Figure 5: We conducted tests on a globally-distributed P2P network and measured (a) the time required to send intent messages and receive confirmations from the validator committee and (b) the time required to disseminate blocks of different sizes to the entire network.**

peer-to-peer. After that, we estimate the latency and throughput in identity aging using the message delivery times.

### 6.1 Experimental Setup

To measure message delivery, we built a globally distributed peer-to-peer network using Amazon’s AWS infrastructure. We instantiated nodes in Frankfurt, London, Singapore, Mumbai and Oregon. We used the EC2 compute services with nodes ranging from t2.micro (single vCPU with 1 GB RAM) to m4.2xlarge (8 vCPUs and 16 GB RAM). The node software was written in Java and run on Ubuntu/Linux OS.

To simulate the worst case scenario in committee selection, we ensured that the miner was never located in the same data center as any of the validators. To simulate global distribution of participants in the peer-to-peer network, we enforced that messages have to travel through at least  $x$  different nodes ( $x$  being 0, 2 and 6) before reaching their destination. We set the intent and confirmation message sizes to 1 KB (although actual messages are smaller). For blocks we tested for three sizes: 500KB, 1MB and 2MB.

We used Dandelion-style message passing [48] that reduces the number of long-distance hops in a global P2P network. In particular, we kept the out-degree of each miner’s connections small but targeted – they connect to a subset of relay nodes that were globally distributed. The relay nodes, in turn, have a large out-degree

for quick transmission within a geographic location. Appendix A provides further details on our peer-to-peer network setup.

### 6.2 Networking Results

Using such network setup, we measured message delivery times for various system and block size. Figure 5 summarizes the results of our experiments. In Figure 5a we the time required for miner selection (combined intent message delivery and confirmation message reception from validator committee). This time grows from 130 ms for small validator committee size  $V = 5$  to 257 ms for large committee size  $V = 1000$ . We conclude that setting the duration of these phases to one second is sufficient in a network environment like ours. We include the arguably excess buffer to account for clock drifts and network jitters. In our testing, we found that setting this value to 300 ms was sufficient, however, we recommend 1 second in recognition of the higher heterogeneity of real-world networks compared to our test environment.

Figure 5b shows the time required for block dissemination (95th percentile figures) that grows from 357 ms for a system size of  $n = 10$  active nodes to 1.1 seconds for a system size of  $n = 10,000$  active nodes. We conclude that setting the duration of block dissemination phase to 4 seconds is sufficient.

We acknowledge that real-world performance might differ from these numbers since the AWS nodes used in our experiments may have better networking connection than that can be expected from ordinary participants. The epoch times would need to be evaluated per network layout and deployment. However, we believe that our set values of 1 and 4 seconds are a good rule of thumb, and they recent estimates for the Ethereum network [11]. In our experiments we observed that Dandelion-like message passing reduced message delivery times significantly which may in part explain why our test network provides better performance than the current Ethereum or Bitcoin networks.

### 6.3 Estimated Throughput and Latency

In Bitcoin, throughput  $tp$  can be approximated as:

$$tp = \frac{\frac{1}{t_r} \times B}{T},$$

where  $B$  is the block size,  $t_r$  is the round time (or block generation rate) and  $T$  is the transaction size.

In our system, the block header grows with the validator committee size due to the confirmation messages that need to be included for tracking currently active nodes. We include enrollment messages to blocks which need to be accounted when estimating throughput. Therefore, in identity aging, throughput can be approximated as:

$$tp = \frac{\frac{1}{t_r} \times (B - H - (V \times S_C) - (n_e \times S_E))}{T},$$

where  $V$  is the validator committee size,  $H$  is invariant block header (128 bytes),  $S_C$  is the size of the confirmation message (416 bytes),  $n_e$  is the average number of enrolled identities per round, and  $S_E$  is the size of the enrollment message.

Assuming a block size of 2MB,  $V = 100$  validators (for  $n = 10,000$  active identities) and only few enrollments per round (owing to the low block time), block header in total consume less than 1%

of the block (i.e., leaving 99%+ for the transaction). Assuming a very large system with  $V = 1000$  validators (for  $n = 100,000$  active identities) and similar block size, the total overhead increases to approximately 8% of the block. In a system of approximately 1.5 million active identities, the block would be almost completely consumed by confirmation messages.

To estimate throughput, we assume an average transactions size of 250 bytes similar to Bitcoin [9]. Assuming a system of 10,000 active identities (approximately the size of the current Bitcoin network), block size of 2MB and 100 validators, identity aging would provide a throughput of approximately 1500 transaction per second. This is two orders of magnitude more than Bitcoin. Latency  $l$  is given by the chosen round duration and assumed fork depth, e.g.,  $l = t_r \times d = 15$  seconds.

We note that given the sub-linear scale-up in block dissemination times versus block size (see in Fig. 5b that quadrupling the block size only doubled the dissemination time), it may be possible to further improve our throughput by experimenting with larger block sizes. We were limited to 2MB blocks due to our test environment cost.

## 7 DISCUSSION

**Other infrastructures.** Since SCIFER does not rely on enclave execution integrity or data confidentiality, our approach is not limited to trusted execution environments, but similar reliable identities could be bootstrapped also from other existing infrastructures that cannot be easily manipulated. For example, mobile phone operators or credit card companies could take the role of IAS attestation service and provide an interface that allows their customers to enroll new identities in controlled manner (e.g., one identity per person or mobile subscription).

Another existing infrastructure that could be leveraged through attestation is TrustZone [6] smartphones. Also the recent efforts to standardize EPID provisioning and attestation across manufacturers [24, 25] could provide a vendor independent way of bootstrapping identities in near future. Identities could be also bootstrapped from multiple sources (say, credit card number and TrustZone smartphone and registered phone number) to provide the right security and usability properties for the particular use case.

**Penalizing malicious behavior.** In most fully-decentralized consensus systems identities can be easily changed (e.g., in Bitcoin the miner could use a different public key every time he starts mining for a new block). In our approach identities cannot be changed after the initial enrollment. One advantage of such long-lived identities is that penalizing malicious behavior becomes possible. For example, if a validator committee member confirms multiple intents on the same round, any entity that observes this can broadcast the conflicting (and signed) confirmation messages and the next miner can include them to a new block as a cheating evidence to eliminate the malicious identity from the system. Thus, users have an incentive to avoid such misbehavior. In systems with changeable identities, similar incentives are typically missing.

**Privacy implications.** In identity aging, mining is based on long-term identities, and therefore correlation of mining events by the same participant becomes trivial. This is a limitation of our approach compared consensus systems where participants pick

new identities for every mining attempt or round. However, we stress that the identities used for transactions can be separate from those used for mining. For example, transactions can use changeable pseudonyms or cryptographic commitments that hide user identities and transaction values [26, 35].

**Network-level attacks.** Another minor drawback of our system is that it can make denial-of-service (DoS) attacks easier. This is because miner candidate selection in our system is predictable, in contrast to unpredictable PoW-mining or systems where a large set is chosen as collective miner (voting systems). To prevent DoS attacks, the participants could attempt to hide their IP addresses. For example, the Dandelion routing [48] that we use in our experiments hides source IP addresses (to some extent).

Eclipsing a participant [23] is less effective in our approach compared to systems like Bitcoin. Although an eclipsed victim's view of the recently mined blocks can be biased, the adversary cannot create sufficiently long forks that would make the victim node accept double spending.

**Stale chips.** In SCIFER, participants are incentivized to buy the cheapest SGX processors that enable enrollment. The market value of a typical CPU decreases substantially in few years after its release. If processors have significantly different value, this could raise questions about the fairness of our approach [7]. We argue that Intel is unlikely to sell unused but outdated products in mass-scale and purchasing cheap second-hand processors does not necessarily provide an advantage either, because those CPUs may have already been enrolled. Also enrollment of old CPUs could be prevented. Recall that the attestation group signature does not identify the individual CPU but it does reveal the manufacturing batch.

**Sharding incorporation.** Our approach provides significantly improved performance over Bitcoin, but because identity aging creates only one block per round our systems cannot provide comparable throughput as sharding based solutions that process transactions in multiple committees in parallel. Also the fact that blocks need to include potentially many confirmation messages to track active identities reduces throughput for very large systems. An interesting direction for future work would be to combine reliable identities established through identity aging (energy efficiency) and parallel processing of transactions through sharding (highest throughput).

## 8 RELATED WORK

In this section we compare prior solutions to our systems primarily based on their efficiency and security. For a more comprehensive comparison or recent blockchain consensus schemes, we refer the reader to [7].

### 8.1 Partially-Decentralized Solutions

Similar to SCIFER, usage of SGX has been proposed for efficient blockchain consensus in few recent works.

**PoET.** Proof of Elapsed Time (PoET) [1] leverages the control-flow integrity of SGX enclaves that wait a randomly-chosen time and the enclaves that finishes waiting first becomes the miner. The obvious drawback of this approach is that a single compromised platform wins on every round. Such majority miner can create

System	Efficiency (compared to Bitcoin [39])				Dependencies	
	Saves energy	Higher throughput	Lower latency	Scalability	Tolerates CPU compromise	Support various infrastructures
Proof of Elapsed Time [1]	•	?	?	×	×	×
Proof of Luck [38]	•	?	?	×	×	×
REM [49]	×	×	×	×	○	×
SCIFER	•	•	•	×	•	•

**Table 1: Comparison of partially-decentralized solutions.**  
**Legend:** fully-provided (•), partially-provided (○), unsupported (×) and unspecified (?).

forks of arbitrary length and reject all transactions from a target victim. While it may be possible to develop heuristics to detect processors that win the waiting game “too often”, compromised processors will always have an advantage over honest ones.

Proof of Luck (PoL) [38] is another similar proposal where each participant enclave picks a random value that is used to choose the miner on the current round.

**REM.** Resource Efficient Mining (REM) [49] replaces the hash computation of Proof of Work with “useful” computation. Each participating enclave performs processing, such as protein unfolding algorithm, and produces an evidence of this computation called Proof of Useful Work (PoUW). REM addresses SGX compromise by limiting the advantage that a compromised CPU can get by capping the accepted rate of useful computation an enclave can report to an estimated maximum value. The main drawback is that continuous computation is needed to maintaining consensus.

**Other.** Teechain [32] uses SGX enclaves to establish secure off-chain payment channels that can improve throughput and latency, but such solutions do not address the problem of blockchain consensus. Also Teechain is vulnerable to CPU compromise.

**Summary.** Table 1 summarizes our comparison. SCIFER is the only SGX solution that saves energy, improves throughput and latency and tolerates compromised CPUs. Additionally, our solution enables bootstrapping identities from different infrastructures besides SGX. None of the solutions is scalable in the sense that systems with more participants provide higher throughput.

## 8.2 Fully-Decentralized Solutions

Similar to DIFER, many recent works have addressed the problem of blockchain consensus in a fully-decentralized setting. Here we review representative solutions.

**Proof of Work.** Bitcoin-NG [20] decouples the two main tasks of Bitcoin mining: leader election and transaction serialization. Every epoch a leader is chosen using Proof of Work. The leader can generate multiple “microblocks” that serialize transactions. Bitcoin-NG increases throughput, but does not save energy. Latency is similar to Bitcoin. FruitChains [42] is another proposal with two types of blocks that improves energy efficiency, but still requires constant computation. Throughput and latency are similar to Bitcoin. In GHOST [46], the heaviest subtree is selected instead of the

longest chain. Similarly, GHOST can reduce wasted blocks, but it still requires constant computation.

In comparison, DIFER requires no constant computation and it improves both throughput and latency. The main drawback is lesser DoS attack resilience and slightly increased messaging.

**Sharding.** Elastico [34] uses sharding to improve blockchain performance. During each epoch, participants establish identities using Proof of Work, all identities in the system are split into committees (shards), each committee agrees consensus on a subset of transactions using a BFT protocol, and a final committee agrees consensus over the output of each shard, again using a BFT protocol. Solidus [5], ByzCoin [30], PeerCensus [15] and Hybrid Consensus [43] are similar designs. OmniLedger [31] enables secure processing of transactions that touch more than one shard.

Sharding enables high throughput, low latency and scalability (i.e., increase in the number of participants increases throughput). The main drawback is that identity creation requires constant computation. DIFER offer smaller throughput and latency improvement compared to sharding, but in turn saves energy. Combining identity aging and sharding is an interesting direction for future work.

**Proof of Stake.** Algorand [37] assigns a weight to each user based on the amount of money they have (i.e., their *stake* in the system). On each epoch, using such weights, a random set of users is chosen to a committee that in turn agrees consensus over transaction using a BFT protocol. Algorand assumes a fully-adaptive adversary and selects the validator committee using cryptographic sortition and a customized consensus protocol that requires several rounds of communication. Similarly in Ouroboros [27] the probability of being chosen to a committee is proportional to owned money. Also Ouroboros considers a fully-adaptive adversary and the committee selection leverages a secure multi-party computation (MPC) that also requires several communication rounds. Show White [14] is a further example of Proof of Stake consensus system.

DIFER provides comparable efficiency as Proof-of-Stake systems that often assume a (weakly) synchronous communication, where as we consider unreliably message delivery which corresponds to measurements from the Bitcoin network [16]. Our system has built-in incentives that are missing from some Proof-of-Stake systems [4] and it also requires fewer messaging rounds and enables penalizing misbehaving participants. Our adversary is not fully-adaptive.

**Proof of X.** In SpaceMint [41], the probability of a node being chosen as a miner is proportional to the amount of storage space that a miner dedicates. The rationale is that doing a small computation over stored data is more energy efficient than Proof of Work. Such solutions do not improve throughput or latency.

**Coin age.** BlackCoin (up to v2) [47] and PeerCoin [28] use the concept of coin aging that has similarities to our identity aging. However, both of these systems are vulnerable to the nothing-at-stake attack and therefore they rely on centralized time-stamping service to ensure that deep forks cannot occur. Also their stake is inherently linked to coins, (i.e., no fungibility) which is a severe shortcoming for a currency system.

**Other.** Sleepy consensus [44] is designed to progress and provide consensus even when the majority of committee members are offline, assuming that among those that are online, the majority behave honestly for a sufficiently long period of time. Our systems track active identities by including confirmation messages to blocks.

System	Efficiency (compared to Bitcoin [39])					Security			
	Saves energy	Higher throughput	Lower latency	Scalability	Message complexity	Adaptive attacker	Allows penalties	Participation incentives	No incentives for pooled mining
<b>Proof of Work</b>									
Bitcoin-NG [20]	×	•	×	×	$O(1)$	•	×	•	×
Fruitchains [42]	○	×	×	×	$O(1)$	•	×	•	×
GHOST [46]	○	×	×	×	$O(1)$	•	×	•	×
<b>Sharding</b>									
Elastico [34]	×	•	•	•	$O(n^2)$	×	×	○	×
Solidus [5]	×	•	•	?	$O(n^2)$	×	×	•	×
ByzCoin [30]	×	•	•	×	$O(n)$	×	?	○	×
OmniLedger [31]	×	×	•	•	$O(n)$	×	×	○	×
<b>Proof of Stake</b>									
Algorand [37]	•	×	•	×	$O(n^2)$	•	×	×	•
Ouroboros [27]	•	•	•	×	$O(nc)$	•	×	•	×
Snow White [14]	×	•	?	•	$O(1)$	×	×	•	×
<b>Proof of X</b>									
SpaceMint [41]	○	×	×	×	$O(1)$	•	×	•	×
SlimCoin [40]	○	×	×	×	$O(1)$	•	×	•	×
<b>Coin Age</b>									
BlackCoin* [47]	•	?	?	×	$O(1)$	•	×	○	×
Peercoin* [28]	•	×	?	×	$O(1)$	•	×	○	×
<b>DIFER</b>	•	•	•	×	$O( c )$	×	•	•	•

**Table 2: Comparison of fully-decentralized solutions. Legend: fully-provided (•), partially-provided (○), unsupported (×), unspecified (?), number of nodes ( $n$ ) and committees ( $c$ ), committee size ( $|c|$ ), centralized components (\*).**

If a majority of participants go offline suddenly, then identity aging cannot provide liveness. Such guarantee requires several rounds of communication per each round.

**Summary.** Table 2 summarizes our comparison. DIFER saves energy and improves Bitcoin’s performance significantly, but does not scale like sharding. Our system requires less messaging than solutions that consider a fully-adaptive adversary. Another advantage of our solution is that it allows penalizing malicious participants. Moreover, since identity aging provides predictable rewards to participants, there is no incentive to join mining pools which effectively centralize many comparable systems.

## 9 CONCLUSION

Bitcoin has shown that it is possible to realize a digital currency in a fully-decentralized setting, but its consensus mechanism has serious shortcomings. In this paper we have proposed identity aging as a new blockchain consensus approach that leverages long-term reliable identities that can be either bootstrapped from an existing infrastructure or mined without any trusted authority. Identity aging saves energy, improves throughput and latency, and provides good security. We believe that our approach can be an important step forward in the quest for better blockchain consensus.

## ACKNOWLEDGEMENTS

The authors would like to thank Ross Anderson, George Danezis and Sheharbano Khattak for their valuable feedback on an earlier draft of this paper. The first author is supported by an academic scholarship from Thales.

## REFERENCES

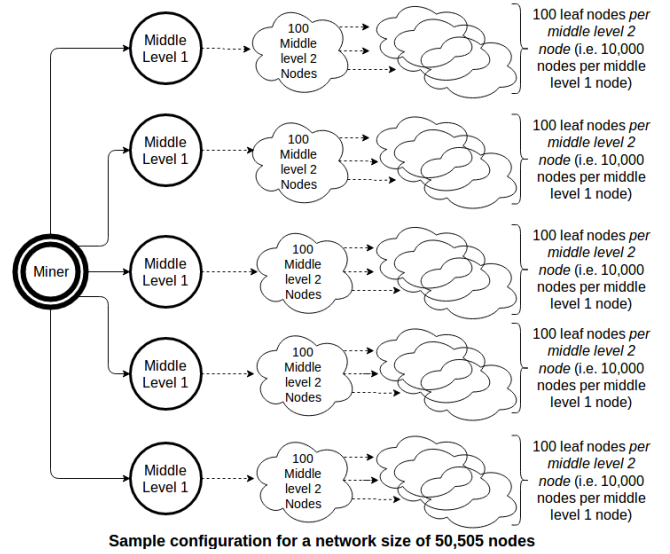
- [1] Sawtooth lake documentation, 2016. <https://intelledger.github.io/introduction.html>.
- [2] Attestation service for intel® software guard extensions (intel sgx): Api documentation, 2017. <https://software.intel.com/sites/default/files/managed/7e/3b/ias-api-spec.pdf>.
- [3] Intel sgx homepage, 2017. <https://software.intel.com/en-us/sgx>.
- [4] No incentive? algorand blockchain sparks debate at cryptography event, 2017. <https://www.coindesk.com/no-incentive-algorand-blockchain-sparks-debate-cryptography-event>.
- [5] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solidus: An Incentive-compatible Cryptocurrency Based on Permissionless Byzantine Consensus. 2016.
- [6] ARM. Trustzone. <https://www.arm.com/products/security-on-arm/trustzone>.
- [7] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.
- [8] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonimisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.
- [9] StackExchange Bitcoin. What is the average size of a bitcoin transaction? <https://bitcoin.stackexchange.com/questions/31974/what-is-the-average-size-of-a-bitcoin-transaction>.
- [10] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.
- [11] Vitalik Buterin. Toward a 12-second block time, 2014. <https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/>.
- [12] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. Sgxpectre attacks: Leaking enclave secrets via speculative execution. *arXiv preprint arXiv:1802.09085*, 2018.
- [13] Victor Costan and Srinivas Devadas. Intel sgx explained. Cryptology ePrint Archive, Report 2016/086, 2016. <http://eprint.iacr.org/2016/086>.
- [14] Phil Daian Rafael Pass Elaine Shi Cornell. Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake.
- [15] Christian Decker, Jochen Seidel, and Roger Wattenhofer. Bitcoin meets strong consistency. *CoRR*, abs/1412.7935, 2014.
- [16] Christian Decker and Roger Wattenhofer. Information Propagation in the Bitcoin Network. In *International Conference on Peer-to-Peer Computing (P2P)*, 2013.
- [17] Digiconomist. Bitcoin energy consumption index. <https://digiconomist.net/bitcoin-energy-consumption>.
- [18] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
- [19] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.
- [20] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. oct 2015.
- [21] Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, pages 281–310, 2015.
- [22] Arthur Gervais, Ghassan Karamé, Srdjan Capkun, and Vedran Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3):54–60, 2014.
- [23] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin’s peer-to-peer network. In *USENIX Security Symposium*, pages 129–144, 2015.
- [24] Intel. Atmel and microchip adopt intel identity technology for iot. [http://download.intel.com/newsroom/kits/idf/2015\\_fall/pdfs/Intel\\_EPID\\_Fact\\_Sheet.pdf](http://download.intel.com/newsroom/kits/idf/2015_fall/pdfs/Intel_EPID_Fact_Sheet.pdf).
- [25] Intel. A cost-effective foundation for end-to-end iot security, 2017. <https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/intel-epid-white-paper.pdf>.
- [26] Tom Elvis Jedusor. Mumblewimble. <http://mumblewimble.org/mumblewimble.txt>.
- [27] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10401 LNCS:357–388, 2017.
- [28] Sunny King and Scott Nadal. Peercoin whitepaper, 2012. <https://peercoin.net/whitepaper>.
- [29] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *ArXiv e-prints*, January 2018.
- [30] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. 2016.
- [31] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger. *IACR Cryptology ePrint Archive*, 2017:406, 2017.

- [32] Joshua Lind, Ittay Eyal, Florian Kelbert, Oded Naor, Peter R. Pietzuch, and Emin Gün Sirer. Teechain: Scalable blockchain payments using trusted execution environments. *CoRR*, abs/1707.05454, 2017.
- [33] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown. *ArXiv e-prints*, January 2018.
- [34] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 17–30, New York, NY, USA, 2016. ACM.
- [35] Gregory Maxwell. Confidential transactions. <https://people.xiph.org/~greg/confidential.values.txt>, 2015.
- [36] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM, 2013.
- [37] Silvio Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016.
- [38] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. Proof of luck: An efficient blockchain consensus protocol. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*, SysTEX '16, pages 2:1–2:6, New York, NY, USA, 2016. ACM.
- [39] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.
- [40] P4Titan. Slimcoin: A peer-to-peer crypto-currency with proof-of-burn, 2014. <https://github.com/slimcoin-project/slimcoin-project.github.io/raw/master/whitepaperSLM.pdf>.
- [41] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. *Cryptology ePrint Archive*, Report 2015/528, 2015. <http://eprint.iacr.org/2015/528>.
- [42] Rafael Pass and Elaine Shi. FruitChains. In *Proceedings of the ACM Symposium on Principles of Distributed Computing - PODC '17*, volume 2016, pages 315–324. ACM Press, 2017.
- [43] Rafael Pass and Elaine Shi. Hybrid Consensus: Scalable Consensus in the Permissionless Model. *31st International Symposium on Distributed Computing (DISC 2017)*, 91:39:1–39:16, 2017.
- [44] Rafael Pass and Elaine Shi. The sleepy model of consensus. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 380–409. Springer, 2017.
- [45] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014.
- [46] Yonatan Sompolsky and Aviv Zohar. *Secure High-Rate Transaction Processing in Bitcoin*, pages 507–527. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
- [47] Pavel Vasin. Blackcoin’s proof-of-stake protocol v2, 2015. <https://bravenewcoin.com/assets/Whitepapers/blackcoin-pos-protocol-v2-whitepaper.pdf>.
- [48] Shaileshh Bojja Venkatakrisnan, Giulia C. Fanti, and Pramod Viswanath. Dandelion: Redesigning the bitcoin network for anonymity. *POMACS*, 1:22:1–22:34, 2017.
- [49] Fan Zhang, Ittay Eyal, Robert Escriva, Ari Juels, and Robbert van Renesse. REM: resource-efficient mining for blockchains. *IACR Cryptology ePrint Archive*, 2017:179, 2017.

## A EXPERIMENTAL NETWORK SETUP

In this appendix we provide further details on our peer-to-peer network test environment.

One of the optimizations that we performed to achieve fast message delivery times in a global P2P network is the implementation of a Dandelion-like networking structure [48]. During our preliminary testing we observed that the bulk of our delay came from the initial block transmission by the miner (due to a high out-degree) and due to multiple hops across geographically distant locations. To solve these issues, we implemented a networking structure as shown in Figure 6. Essentially, we elected some nodes within a cluster of geographically close nodes to serve as “Middle clients” – nodes that are directly connected to by miners when choosing to broadcast the block (we did not use this in the intent phase as it did not lead to any noticeable improvement in performance). These middle clients could then afford to have a large out-degree owing



**Figure 6: Overview of the networking layout.** The different labels for the nodes are for illustrative purposes only – all the nodes run identical software. The core optimization here is the introduction of well-known “Middle Clients” that have a large out-degree in a particular geographical cluster, thus reducing the number of long-distance hops required.

to their proximity to a large number of nodes. This optimization led to a significant reduction in latency.

We note that the middle clients are in no way different from all the other nodes. Any node could be chosen as a middle client (they all run identical software) and messages may be broadcast to multiple middle clients within a cluster. We imagine that in a large community-backed deployment, these middle clients may end up being chosen by some reliability and performance metrics as seen in the Tor network. We recommend that maintainers of existing consensus frameworks should explore similar geographically-minded networking schemes as it could lead to substantial improvements in performance.

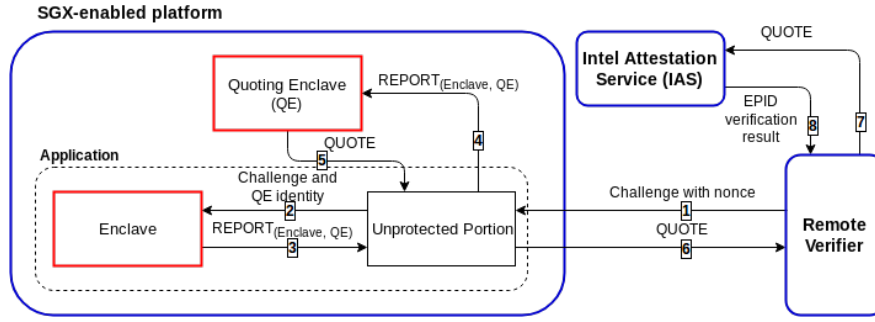
## B INTEL SGX BACKGROUND

In this appendix we provide brief background information on Intel SGX [3]. For a more detailed explanation of the architecture, we refer the reader to [13].

### B.1 SGX Architecture

Intel’s Software Guard Extension (SGX) introduces a set of CPU instructions for creating and managing protected applications called *enclaves*. Enclaves are isolated from all software running on the system, including privileged software like the OS [13]. The SGX trust model assumes the CPU itself to be the only trustworthy hardware component of the system, i.e., enclave data is handled in plain-text only inside the CPU. Whenever data is moved out of the CPU, e.g., into the memory (DRAM), it is encrypted and integrity protected using a dedicated memory encryption engine in the CPU.





**Figure 7: SGX attestation overview. Remote attestation is an interactive protocol that involves three parties: remote verifier, the attested SGX platform and Intel Attestation Service (IAS).**

The OS, although untrusted, is responsible for creating and managing enclaves. The operating system allocates memory for the enclaves and copies the initial data and code into the enclave. Initialization actions of the OS are recorded securely by SGX inside the CPU. The initialization process creates a *measurement* that captures the enclave’s code configuration and can be used for later verification by an external party (attestation). The *sealing* capability of SGX enables persistent secure storage of enclave data such that the data is only available to correctly created instances of the same enclave that originally saved it.

Enclaves cannot directly execute system calls, and therefore developers must divide their applications into two parts. Protected processing takes place within the enclave and an unprotected part (run as normal user-level process) must handle operations such as file system access and networking using the untrusted OS.

## B.2 SGX Remote Attestation

Remote attestation is a procedure, where an external verifier checks that certain enclave code is correctly initialized and running on the attested device. Remote attestation is an interactive protocol between three parties: (i) the remote verifier, (ii) the attested SGX platform, and (iii) Intel Attestation Service (IAS), an online service operated by Intel.

During manufacturing each SGX processor is equipped with a unique attestation key that IAS uses for verification. Each SGX platform includes a trusted system component called Quoting Enclave that has exclusive access to the attestation key.

The remote attestation process is illustrated in Figure 7: (1) The remote verifier sends a random challenge to the attested platform. (2) The unprotected application forwards the challenge to the enclave

that (3) returns a REPORT data structure encrypted for the Quoting Enclave containing the enclave’s measurement created during its initialization. The REPORT data structure includes a USERDATA field, where the attested enclave can include application-specific attestation information, e.g., hash of public key. (4) The unprotected application forwards REPORT to Quoting Enclave that (5) verifies it and returns a QUOTE structure signed by the processor-specific attestation key. (6) The unprotected application sends QUOTE to the remote verifier that (7) forwards it to the IAS online service using TLS. (8) IAS verifies the QUOTE signature, checks that the attestation key has not been revoked, and in case of successful attestation returns the QUOTE structure signed by IAS. A successful protocol run also establishes a secure channel between the remote verifier and the enclave.

The attestation key is a part of a group signature scheme called EPID (Enhanced Privacy ID) [25] that supports two signature modes. The default mode is privacy-preserving and does not uniquely identify the processor to IAS (the signature only identifies a group like certain processor manufacturing batch). Linkable signature mode allows IAS to verify, if the currently attested CPU is the same as previously attested CPU.

Usage of SGX attestation requires registration with Intel. In Intel terminology, the attestation verifier is called “service provider”, and upon registration each service provider receives a credential that they use to authenticate to IAS. Linkable mode of attestation needs to be enabled at the time of service provider registration. If linkable mode of attestation is used, IAS reports the same *pseudonym* every time the same service provider requests attestation of the same CPU [2].