GreenCoin: A Renewable Energy-Aware Cryptocurrency

Nazmus Saquib, Shivaansh Kapoor, Chandra Krintz, Rich Wolski
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, USA
{nazmus, shivaansh, ckrintz, rich}@cs.ucsb.edu

Markus Mock
Department of Computer Science
University of Applied Sciences
Landshut, Germany
mock@haw-landshut.de

***Abstract*—In this paper, we propose GreenCoin – an energy-efficient cryptocurrency system with mining protocols designed to favor locations with relatively higher availability of renewable energy. Traditionally, crypto coin mining involves solving complex mathematical problems by high-end computing devices consuming an enormous amount of electricity, thus adversely affecting net carbon emissions. To reduce cost and emissions, GreenCoin uses a modified proof of stake (PoS) consensus algorithm, which itself is more energy efficient compared to other state-of-the-art methods. Our modified PoS algorithm, called Green PoS (GPoS), allows GreenCoin to favor nodes (with reward and privilege) located in regions with higher availability of renewable energy. We present a detailed system architecture of GreenCoin and explain the operating method of GPoS. We also provide results from empirical studies demonstrating the renewable energy-aware approach of GreenCoin.**

***Index Terms*—renewable energy, blockchain, cryptocurrency**

I. Introduction

Cryptocurrencies are a set of technological innovations that have the potential to revolutionize society along several dimensions, from commerce to political and corporate governance, to social trust. These innovations and potential benefits carry environmental costs largely (but not exclusively) accruing to the energy required by distributed cryptocurrency implementations. At a high level, a "crypto coin" (an unforgeable value-carrying token) represents some amount of "work" associated with its production.

In the original formulations of cryptocurrencies (cf. [1], [2]) "miners" solved computationally intensive or memory intensive [3] puzzles, the solutions they announced to all participants, which could easily verify the solutions. These "proof-of-work" (PoW) protocols created a consensus among all participants about the temporal order in which puzzles were solved that could only be subverted by an adversary controlling a substantial fraction [4] of all of the participants. By attaching a transaction to each puzzle solution, then, the overall system could achieve consensus on transaction order in a way that is difficult (or expensive) to subvert. In particular, it was possible to determine the "earliest" transaction that achieved consensus thereby preventing one puzzle from being used to represent multiple transactions. That is, the solution to a puzzle (termed the "minting" of a crypto coin), could only be spent once.

PoW protocols are currently in use by many cryptocurrencies, e.g., Dogecoin [5], Monero [6], Litecoin [3], and most notably Bitcoin [1]. However, they all require the computers solving the puzzles (i.e., the "miners") to expend much electrical energy on each puzzle solution. Furthermore, puzzle solutions are deliberately made rare to prevent inflation. Thus as solver technology improves (often due to greater energy expenditure), energy consumption by miners increases.

To address the environmental concerns associated with PoW protocols, particularly concerning the use of fossil fuels to generate the electricity they require, cryptocurrencies have been turning to new "Proof-of-Stake" (PoS) protocols that are far more energy efficient. PoS protocols, unlike their PoW counterparts, rely on economic penalties to incentivize truth-telling concerning transaction order and uniqueness. With these new protocols, "miners" announce transactions without solving a puzzle associated with each. Instead, each miner is "staked" to a hoard of crypto cash (that carries value) which can be confiscated should a miner make a false announcement where veracity is determined by consensus.

By eliminating the need to solve energy-consuming puzzles, PoS protocols are far more energy efficient than corresponding PoW protocols. As a result, many current cryptocurrencies (c.f. [7]–[9]) have begun to use them, most notably Ethereum [10].

Regardless of whether a cryptocurrency uses a PoW protocol or a PoS protocol, however, they require a distributed collection of computers to achieve consensus on a global transaction order in a way that can be recorded (again by all participants) that is tamper-proof. These computers must consume electrical power to implement the protocol functionality.

In this paper, we describe GreenCoin – a cryptocurrency that marks each crypto coin with an indelible measure of the renewable energy that was used to create it. GreenCoin relies on a new PoS protocol (GPoS) to attach a "green score" (that maps to renewable energy available at the time of the transaction proposal) to each crypto coin in a way that cannot be removed. Further, GreenCoin and GPoS permit the formulation of "green" smart contracts that specify a minimum greenness score that must be presented to execute the contract. Note that we have defined GPoS as a PoS protocol as we believe future protocols are likely to take this form, but the fundamental tenets of GreenCoin apply to PoW protocols as well.

Implementing GreenCoin and GPoS requires that we

- develop a way to score each coin with a greenness score and to compose/decompose scores from multiple coins when they are used to represent transactions,
- develop an incentive system that favors (i.e., attaches greater value to) greener coins, and
- develop a system for securely determining the amount of renewable energy used when minting a coin.

In this paper, we describe research results addressing these three challenges. We describe a methodology for attaching a green score to each coin, each wallet (containing multiple coins), and for transacting green scores. We also describe the use of Trusted Execution Environments (TEEs) and attestation [11] to determine the location of each machine proposing a transaction. This location information then indexes fuel mix data by location either from a publicly available database such as the US Energy Information Administration (EIA) database [12] or from energy utility records that must be published in the ledger. In this work, we assume that fuel mix data by geographic region is available in real-time (or near real-time) from a trusted data source. We validate these results using a prototype "permissioned" implementation of GreenCoin designed to support Internet of Things (IoT) deployments comprising participants whose identities are not anonymized. Our results indicate that GreenCoin feasibly captures and publishes into the ledger the relative renewable energy usage associated with each blockchain transaction and also enables smart contracts that are contingent upon sufficient use of renewable energy.

II. Background and Related Work

In this section, we provide a literature review on the basic building blocks of GreenCoin. GreenCoin requires (i) an energy-efficient consensus protocol for blockchain, (ii) a trusted and secure way to find the location of a blockchain node, and (iii) a method to incorporate a trusted source of fuel mix data for use by the system. Hence, we first compare different protocols used in current cryptocurrency systems and explain our choice of PoS. Next, we discuss the current state-of-the-art in location verification and justify our choice of using trusted execution environments. Finally, we present sources of fuel mix data, review methods to incorporate third-party data into blockchain, and introduce our chosen system to do so, called Depot. Depot is an open-source data lake that is designed to allow community data contribution with contributor-defined access control policies and shared hosting costs. Depot was developed as part of the RiPiT project (cf. [13], [14]) to host carbon-emissions data from public sources and community-contributing researchers.

A. Consensus Protocols

Most cryptocurrencies use blockchain as the core underlying technology [15]. When it comes to blockchain, there are multiple consensus protocols, such as proof of work (PoW), proof of stake (PoS), proof of authority (PoA), proof of retrievability (PoR), proof of elapsed time (PoET), etc. [16]. However, two of the most widely used protocols are PoW and PoS [17].

In PoW, any node that wants to participate in mining, i.e., block generation must solve a computationally complex problem to ensure the validity of the newly mined block [18]. Although finding the solution to this problem is challenging, verifying the solution is easy. The main criticism against PoW is its high energy consumption required to solve these crypto puzzles. The Bitcoin network, which uses PoW, was estimated to consume 2.55 gigawatts of electricity in 2018 [19]. This consumption was projected to reach 7.67 gigawatts, making it comparable to the energy consumption of entire countries such as Ireland (3.1 gigawatts) and Austria (8.2 gigawatts) [19]. As of July 2021, Bitcoin's carbon footprint showed 64.18 megatons of CO_2 emission, close to the emissions by Greece and Oman [20]. This high CO_2 footprint has led to a spike in research interest in energy-efficient protocols.

In contrast, PoS does not involve computationally intensive crypto puzzles and hence it is comparatively energy-efficient. In PoS, a blockchain node can opt to stake, i.e., set aside a portion of its coins as collateral. Instead of mining power as in PoW, the probability of a node being selected as the proposer, i.e., the entity permitted to create the next block (and hence earn the associated reward), is proportional to the stake. Just as a node can receive a reward for honest behavior, it can be penalized for malicious behavior. This penalty is executed in the form of *slashing*, i.e., taking away a portion of the stake. Due to the energy efficiency of PoS over PoW, we base our protocol, Green PoS (GPoS), on PoS.

B. Location Verification

Location-aware Internet applications commonly use IP addresses to determine the location of a connected device [21]. There are two primary methods of IP geolocation: (i) IP geolocation databases and (ii) active network measurements [22].

IP geolocation databases provide a mapping between an IP address and *(lat, long)* coordinates [23]. These databases can be either proprietary [24] or public [25], [26]. Although the exact methods of constructing these databases are not always divulged to the public, they are often based on a combination of *whois* services, autonomous system (AS) numbers, DNS LOC records, etc. [22], [23]. While these databases can achieve country-level accuracy, discrepancies among databases are prevalent regarding city-level accuracy [27]. The accuracy of these databases starts diminishing with increasing granularity. As the availability of renewable energy can vary even within the same city, IP geolocation databases are inadequate for supporting GreenCoins.

Measurement-based geolocation algorithms heavily depend on a set of geographically distributed *landmarks* with known locations [21], [22]. These landmarks measure different network properties between them and the target IP, such as the delay and path taken by traffic [28], [29]. However, the availability of such landmarks can be sparse [21], resulting in lower geolocation accuracy. Moreover, studies suggest that an adversarial target can falsify measurements without detection, thus advertising itself to be at a different location than it truly is [22]. Hence, measurement-based geolocation algorithms are also insufficient for our application.

A relatively accurate approach to determining the location of a machine is to equip it with a GPS module [30]. However, having a tamper-proof GPS module does not prevent a machine from falsifying its location. In addition, we must ensure that both the process that retrieves data from the GPS device and the communication channel between the process and the GPS device are tamper-proof. A prime candidate for such a system is a trusted execution environment (TEE). Studies suggest that using TEEs coupled with GPS devices can provide a secure and tamper-proof way of location verification (cf. [31]–[33]).

Trusted Execution Environments: A *trusted execution environment* is an isolated processing environment in which applications can be executed while precluding malicious interventions of the host OS [34]. Examples of popular technologies used to provide a TEE are Arm TrustZone [35] and Intel SGX [36]. A TEE provides isolation for programs from the rest of the device, called the rich execution environment (REE). REEs typically include an operating system, e.g., Linux, and user space applications. The TEE uses a combination of software and hardware-based security mechanisms to ensure that applications running inside the TEE remain secure even when the REE is compromised [37]. Figure 1a shows a high-level diagram of the interaction among REE, TEE, and peripherals within a single device. Desired security features of a TEE include isolated execution, secure storage, remote attestation, secure provisioning, and trusted path [38]. Many TEEs provide most of these features, while some can be built upon the existing functionalities provided by the TEE. For example, Intel SGX provides built-in remote attestation, which verifies three things: (i) the identity of an application, (ii) whether it is intact, i.e., that it has not been tampered with, and (iii) that it is running securely inside the TEE [39]. Although Intel SGX does not provide a trusted path between peripherals and itself out-of-the-box, multiple studies show it is possible to establish secure paths between I/O devices and Intel SGX [40], [41].

Security Concerns of TEEs and Countermeasures: Despite the promising security features of TEEs, multiple studies have demonstrated that TEEs are susceptible to some attacks [42], [43]. Many of these are side-channel attacks that compromise the secure keys used for attestation, rendering the software running in these TEEs untrustworthy [37]. Cache-based side-channel attacks such as prime+probe [44] and flush+reload [45] observe the timing differences of different measurements to identify whether data is retrieved from the cache or the main memory, thereby allowing the attacker to learn about the memory access patterns of the victim. Transient execution-based side-channel attacks exploit branch misprediction leading to discarded instructions after a pipeline flush [46]. SgxPectre [47] and Foreshadow [48] are examples of transient execution-based side-channel attacks capable of extracting secret keys from the victim TEE. One study suggests using a secure co-processor such as Google's Titan M [49] accessible by TEE but separate from the main processor can circumvent side-channel attacks [37]. The study assumes a secure communication channel between the TEE and the co-processor, which makes the setup free of side-channel attacks. The co-processor is entrusted with performing sensitive cryptographic operations and storing cryptographic secrets.

C. Energy Mix Data

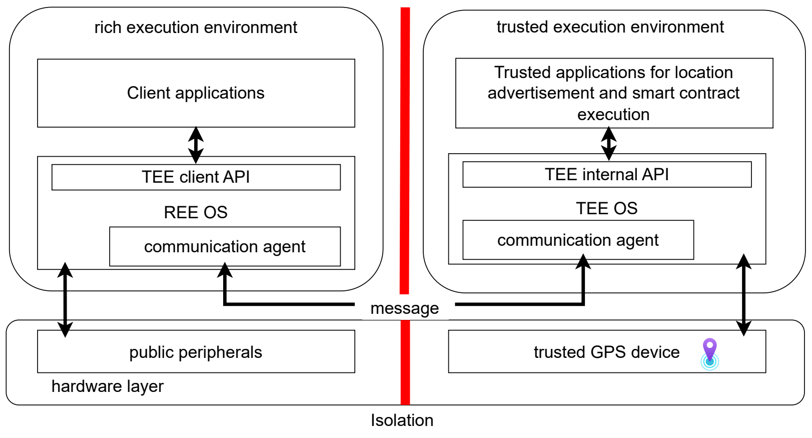
The total energy consumed in a given geographical region can be broken down by primary energy sources. This mapping between what fraction of consumed energy is generated by which source is known as fuel mix or energy mix [50]. Throughout this paper, we refer to the renewable energy portion of the energy mix as the *energy mix score*. Thus the energy mix score can be between 0.0 (no fraction of energy was obtained from renewable sources) to 1.0 (all consumed energy was obtained from renewable sources). This is essentially a greenness score of a region at a given time. The higher this score is, the more green the region is, i.e., the proportionally more the region has energy generated by renewable sources.

Independent System Operations (ISOs) maintain renewable energy data for different regions, e.g., California ISO [51] maintains renewable energy data for California. We can query these data sources to get an estimate of renewable energy that was available at a specific time and region. Hence, when minting a coin in GreenCoin the proposer can query these sources and assign a score to the minted coin. GreenCoin uses an existing system called Depot that pools multiple ISO data sources and exposes an API to query the sources. Note that traditionally, third-party data is incorporated into blockchain networks using oracles [52]. Due to the absence of a competing system providing data similar to Depot, Depot itself acts as an oracle. In Section III, we describe how Depot and a blockchain node securely communicate with each other.

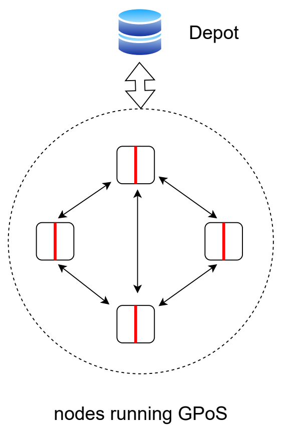
III. GreenCoin System Architecture

In GreenCoin, the location of a blockchain node, i.e., a machine running the blockchain protocol, plays a significant role. The system favors accounts maintained by nodes with a higher fuel mix score when it comes to endowing rewards, e.g., for block creation, and privileges, e.g., executing smart contracts. Hence, there is a strong coupling between a blockchain node and a blockchain account. Although a blockchain node can maintain multiple different accounts, for the simplicity of exposition, we assume there is a one-to-one mapping between a blockchain node and a blockchain account. Throughout this paper, we use the terms node and account interchangeably.

Figure 1 shows the high-level system architecture for GreenCoin using Green PoS (GPoS). As described in Section II-B, each node runs a TEE. Each node is also equipped with a tamper-proof GPS device. This GPS device is connected to the TEE through a secure I/O channel. Blockchain nodes



(a) Details of a single node running a TEE.



(b) Interaction between nodes and Depot.

Fig. 1: High level system architecture of GreenCoin.

contact Depot to retrieve their fuel mix score. As TEEs can be resource constrained, we opt to run only functions which need an extra layer of security and trust on TEEs rather than the full blockchain system. There are two cases when we execute operations inside a TEE: for (i) block proposal, and (ii) smart contract execution.

At the time of block creation, a node must determine the fuel mix score for its location. The location data must come from the TEE to ensure the node cannot advertise a false location. Therefore, to retrieve the fuel mix score, the client application first requests the trusted application running inside TEE to read the location data. In response, the trusted application reads the location from the tamper-proof GPS device through the secure I/O provided by the TEE, signs the location reading, and sends it back to the client application. The client application itself cannot tamper with this location data as the TEE signs it with a tamper-proof hardware key. Once it receives the location data, it sends a request encapsulating the location data to Depot to retrieve the fuel mix score. Depot can check the trustworthiness of the trusted application through remote attestation. Once Depot verifies the program run by TEE, it decrypts the location data and sends the signed fuel mix score to the client application. Upon receipt of the fuel mix score, the client application finalizes the block by adding the signed fuel mix score to the block and propagating it to other nodes. The other nodes do not need to execute any extra verification steps, as they can simply decrypt the signed fuel mix score. Figure 2 delineates the steps involved in retrieving the fuel mix score as a part of the block creation process.

Smart contracts in GreenCoin are also executed inside TEEs. GreenCoin imposes single-node execution of a smart contract (cf. Section IV-F), where the smart contract is executed only by the block proposer, which proposes the block containing the transaction to execute that contract. Hence other nodes must have a mechanism to verify that the proposer executed the intended contract along with the appropriate input and output. GreenCoin uses TEEs to implement this trust mechanism. Whenever a proposed block contains a transaction to execute a contract, the proposer executes the contract in the TEE and propagates supplementary information along with the proposed block to other nodes. The supplementary data includes a verifiable report of the code executed by the TEE, along with its input and output.

In this work, our primary focus is on the underlying blockchain protocol and not the inner workings of off-the-shelf TEEs. Hence, we assume the nodes are running TEEs and implement a proof of concept distributed system executing GPoS using Python. We empirically evaluate GPoS-driven GreenCoin using multiple cloud instances and present our findings in Section V.

IV. GPoS Implementation

This section describes the details of the GPoS building blocks. We first present how coins are represented and how their greenness score is determined. Then, we define the greenness score of a collection of coins, i.e., the wallet or balance of an account. Next, we describe how stakes in GPoS differ from other PoS protocols. Following that, we present how honest proposers are incentivized through block reward and malicious proposers are disincentivized through slashing. Then, we explain the different aspects of a smart contract. Finally, we wrap up this section with an overview of the cryptoeconomics of GPoS-powered GreenCoin.

A. *Coins*

Each coin in GreenCoin is tagged with a greenness score, which signifies the percentage of renewable energy available during the time and at the site of coin generation. Hence, in contrast to other cryptocurrencies, each coin in our system is represented by two values – one to represent the greenness score of the coin and the other to represent the amount of coin. Throughout this paper, we use the tuple $(score, amount)$ to represent a coin. For example, $(0.5, 10.0)$ represents a coin of amount 10.0 having a score of 0.5. The amount of a coin is divisible, but not the score. For example, the owner of the above coin can decide to transfer half the amount of the coin to another account. In this case, the second account receives

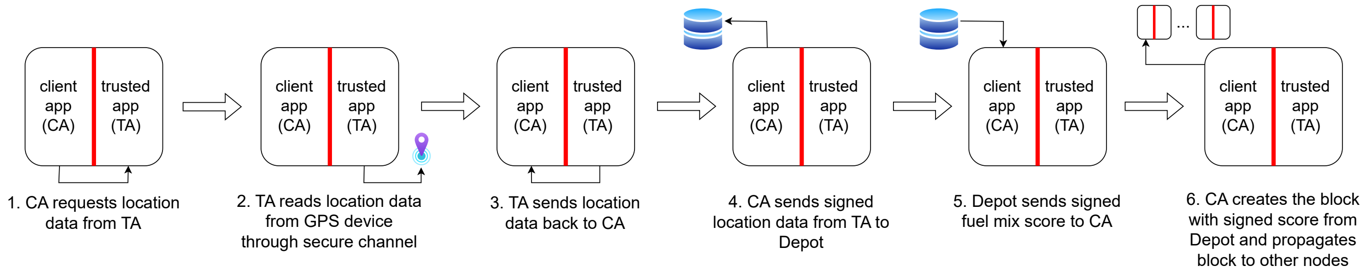

client app (CA)
trusted app (TA)
1. CA requests location data from TA
2. TA reads location data from GPS device through secure channel
3. TA sends location data back to CA
4. CA sends signed location data from TA to Depot
5. Depot sends signed fuel mix score to CA
6. CA creates the block with signed score from Depot and propagates block to other nodes
...

Fig. 2: Steps involved in retrieving fuel mix score for block creation.

the coin $(0.5, 5.0)$ and the first account is left with $(0.5, 5.0)$. However, the first account cannot send $(0.25, 10.0)$ to the second account.

B. Wallet Score

Algorithm 1 getWalletScore

```
Input: list of coins in $(score, amount)$ tuple format, $C$
Output: wallet score
 1: $sumDot \leftarrow 0; sumAmount \leftarrow 0; N \leftarrow$ LENGTH$(C)$
 2: for $k \leftarrow 1$ to $N$ do
 3:     $c \leftarrow C[k]$
 4:     $sumDot \leftarrow sumDot + c.score \times c.amount$
 5:     $sumAmount \leftarrow sumAmount + c.amount$
 6: end for
 7: if $log(sumAmount) \leq 0$ or $log(sumDot) \leq 0$ then
 8:     return 0
 9: else
10:     return $log(sumDot)/log(sumAmount)$
11: end if
```

Over time, an account may consist of an array of coins with different greenness scores. We term the collection of coins owned by an account as its *account balance* or *wallet* and represent it as a list of coins $[(score_1, amount_1), \ldots, (score_n, amount_n)]$. As our goal is to favor accounts having greener coins, we need a metric to evaluate the overall greenness score of an account, which we term the *wallet score*. We define *wallet score* as the ratio of the logarithm of the dot product between the scores and amounts to the logarithm of the sum of amounts. Algorithm 1 shows the wallet score calculation. Note that our choice of wallet score calculation over a normalized dot product allows us to assign different scores for wallets having coins with the same score but different amounts. For example, two different wallets $[(0.5, 10.0)]$ and $[(0.5, 100.0)]$ have a score of 0.50 according to the normalized dot product. However, our approach differentiates these two cases and assigns the scores 0.70 and 0.84 to the two wallets, respectively. Our method, as evident from line 7 of Algorithm 1, uses only non-negative wallet scores setting the value to 0 when either logarithm is negative.

C. Stake

In PoS protocols, an account can "stake" or set aside a portion of its balance as collateral. The higher the stake compared to other accounts, the greater the probability of the account becoming a proposer. Malicious behavior by the account results in a portion of the stake being lost, i.e., slashed (cf. Section IV-E). As our goal is to favor accounts with higher greenness scores, stake in GPoS is not dependent only on the amount of coins owned by an account but on the overall wallet score of the account. However, an account can not selectively stake its balances, as this can give a false representation of the greenness of an account. For example, the wallet score of an account with balance $[(0.1, 1000), (1.0, 10)]$ is 0.68. If the account had been able to stake a portion of its balance, it could have chosen to stake $[(1.0, 10)]$ and advertise itself as having a greenness score of 1.0. Hence, GPoS does not allow an account to explicitly stake balances, instead, the full wallet acts as the stake. Consequently, slashing impacts the full wallet of the account rather than a portion of it.

D. Block Reward

Algorithm 2 endowBlockReward

```
Input: block with embedded signed fuel mix score from Depot, $b$
    minimum reward amount for empty transactions, $m$
Output: none, ensures appropriate state change
 1: $amount \leftarrow 0; N \leftarrow$ LENGTH$(b.transactions)$
 2: for $k \leftarrow 1$ to $N$ do
 3:     $tx \leftarrow b.transactions[k]$
 4:     $amount \leftarrow amount + tx.score$
 5: end for
 6: $amount \leftarrow amount/N$
 7: if $amount = 0$ then
 8:     $amount \leftarrow m$
 9: end if
10: UPDATEBALANCE$(b.proposer, b.depotScore, amount)$    ▷
    adds the coin $(b.depotScore, amount)$ to the block proposer's
    balance, followed by updating its wallet score.
```

A proposer generates a coin c as a reward when a block is successfully proposed with approval from at least 51% of the nodes. The score of c is the fuel mix data received from Depot. This data represents the fraction of energy that is renewable at the time of generation of the coin and the location of the node on which the account is running. The amount of c is the average score of the transactions within the proposed block. Just as a coin has a score, so does a transaction. The score of a transaction depends on its type as presented in Table I. Making the amount of c dependent on the scores of the transactions incentivizes the proposers to include transactions with higher scores in blocks, which in turn encourages all the nodes to maintain a high greenness

TABLE I: Different types of transactions and their scores.

type	description	score
TRANSFER	transfers a coin c from one account to another	score of c
SLASH	slashes an account	predefined default
PEN_SLOW	penalizes slow node	predefined default
CREATE_SC	creates a smart contract	product of $score$ and $amount$ of gas[1]
EXEC_SC	executes a smart contract	product of $score$ and $amount$ of gas[2]

[1,2] cf. Section IV-F

score throughout the GreenCoin deployment. If the proposed block contains an empty transaction list, a predefined default value is used as the amount of c. Algorithm 2 presents the calculation of block reward.

E. Slashing

Algorithm 3 slashAccount

```
Input: public key of slasher, $p_{slasher}$
    public key of slashee, $p_{slashee}$
    list of coins in $(score, amount)$ tuple format owned by slashee, $C_{slashee}$
    minimum amount of coin at or below which the amount is set to 0, $m$
    factor by which amount of coin is reduced at each slashing step, $f$, $(0.0 \leq f < 1.0)$
Output: none, ensures appropriate state change
 1: SORT($C_{slashee}$)    ▷ sorts $C_{slashee}$ in descending order of score
 2: $S_c \leftarrow$ GETWALLETSCORE($C_{slashee}$)    ▷ current wallet score
 3: $S_t \leftarrow S_c \times 0.5$    ▷ target wallet score
 4: $N \leftarrow$LENGTH($C_{slashee}$)
 5: $C_{initial} \leftarrow$COPY($C_{slashee}$)    ▷ creates a copy of $C_{slashee}$
 6: $C_{diff} \leftarrow []$    ▷ list of slashed coins is populated here
 7: for $k \leftarrow 1$ to $N$ do
 8:     while $S_c > S_t$ and $C_{slashee}[k].amount > 0$ do
 9:         if $C_{slashee}[k].amount < m$ then
10:             $C_{slashee}[k].amount \leftarrow 0$
11:         else
12:             $C_{slashee}[k].amount \leftarrow C_{slashee}[k].amount \times f$
13:         end if
14:         $S_c \leftarrow$ GETWALLETSCORE($C_{slashee}$)
15:     end while
16:     $amount' \leftarrow C_{initial}[k].amount - C_{slashee}[k].amount$
17:     APPEND($C_{diff}$, $(C_{slashee}[k].score, amount')$) ▷ appends a coin to the list $C_{diff}$
18:     if $S_c \leq S_t$ then
19:         break
20:     end if
21: end for
22: ASSIGNBALANCES($p_{slashee}, C_{slashee}$)    ▷ sets the balance of slashee to $C_{slashee}$, followed by updating its wallet score
23: ADDBALANCES($p_{slasher}, C_{diff}$)    ▷ adds the coins in $C_{diff}$ to the balance of slasher, followed by updating its wallet score
```

In PoS algorithms, slashing is used to discourage malicious behavior by taking away a portion of the stake of the slashee, i.e., the node being slashed. At the same time, by rewarding a portion of the stake to the slasher, i.e., the block proposer whose block included the slashing transaction, nodes are encouraged to punish malicious behavior actively. In GPoS, slashing performs a similar functionality. However, slashing is relatively complex in GPoS due to the coin being not only a single value but an amount tagged by a score. GPoS slashes a faulty proposer so that its wallet score falls below half its pre-slashed value. Potentially numerous combinations of coins can be taken away to achieve this condition. However, GPoS starts slashing by taking away the coins with higher scores first. At each iteration of slashing, GPoS takes away a portion of the amount of the highest-scored coin and calculates the new wallet score. It stops if the new wallet score is half of the initial score. Otherwise, it continues slashing a portion of the highest-scored coin. If the amount of this coin falls to zero or a predefined minimum value, this coin is completely removed from the account, and the process continues from the next highest coin. Once the set of coins that must be deducted from the malicious node's wallet is determined, those coins are added to the wallet of the slasher as a reward. Algorithm 3 delineates the steps in slashing.

F. Smart Contract

Smart contracts are an integral part of any general-purpose blockchain which intends to extend beyond the application of cryptocurrency. Traditionally, a smart contract is an executable piece of code that is immutable once created and executes on every node when invoked. This is extremely wasteful but imperative when contract execution cannot be verified by other nodes in the system. Conversely, since GreenCoin aims to be an energy-aware blockchain and stipulates that each node is equipped with a TEE, GreenCoin is capable of imposing a single-node execution model. In this, the account which invokes a smart contract can set a minimum threshold for the executor's energy mix score, and a block proposer can only add the invocation transaction to a block if their energy mix score is greater than or equal to the specified threshold. As discussed in Section III, the block proposer sends supplementary data containing a verifiable report of the code executed by the proposer's TEE along with its input and output during block propagation. The other nodes in the system can verify the execution of the smart contract along with necessary state changes from this supplementary data.

Algorithm 4 getMaxExecutionTime

```
Input: gas in $(score, amount)$ format, $g$
    upper range of execution time, $t_{upper}$
    lower range of execution time, $t_{lower}$
Output: maximum execution time in seconds
 1: if $g.score \times g.amount \leq 1$ then
 2:     return $t_{lower}$
 3: end if
 4: $n = log(g.score \times g.amount)$
 5: $d = log(g.amount)$
 6: return $\lceil t_{upper} \times n/d \rceil$
```

Since the languages used to write smart contracts are expressive enough to be Turing complete, they are susceptible to the halting problem. Therefore, smart contract systems generally use the concept of *gas* to ensure termination. Gas refers to the unit of measurement for the computational effort required to execute a particular operation or transaction. If a node executing a smart contract runs out of gas before the computation finishes, the contract execution stops. Since only a finite amount of gas is made available to nodes, the execution is guaranteed to terminate.

In GreenCoin, Gas is an up-front remittance paid by a node R which requests the execution of a smart contract. This is a fee paid by R for using resources on the executor's machine. Gas is represented as a single coin $(score, amount)$. One unit of gas is essentially a coin that is burnt, i.e., not paid to any particular entity but removed from the wallet of R. At the time of execution, gas is used to calculate the CPU time allotted to the contract for execution.

Algorithm 4 shows how we determine the amount of time that can be purchased with a unit of gas. This procedure accepts a predefined range of time $[t_{lower}, t_{upper}]$ and returns a time within this range that can be purchased per unit of gas. If an execution runs out of the time that it paid for, the execution is halted and any state changes that were made are reverted. Given two different units of gas with the same value of $(score \times amount)$, the unit with a higher $score$ component yields a longer execution time according to Algorithm 4. For example, consider the case in which $[t_{lower}, t_{upper}]$ is $[1, 10]$ and compare gas $(0.5, 10)$ to gas $(0.8, 6.25)$. Both gas tuples generate the same product of amount and score but the first can afford a maximum execution time of 7 seconds while the latter can afford a maximum execution time of 9 seconds, when computed on line 6 of Algorithm 4. Note that the conditional statement in line 1 of Algorithm 4 imposes a natural cutoff for the combination of $score$ and $amount$ below which the unit can buy the minimum execution time only.

In GreenCoin, contracts are effectively accounts, known as contract accounts, controlled by the logic in their code rather than by cryptographic key pairs. Contracts can receive coins and send coins programmatically from accounts and contract accounts. Contracts also have access to some additional information: account balances, contract account balances, the states of all contracts, and the invoking transaction's details. This allows contracts to act dynamically based on the current state of the blockchain.

G. Cryptoeconomics

GreenCoin's principal cryptoeconomic goal is to provide incentives that encourage nodes to use renewable energy for every interaction in the system. We provide four key incentives: (i) Given any two accounts with the same amount of coins, the account having coins with a higher score has a greater probability of proposing blocks and generating coins. (ii) Block rewards are generated based on the proposer's energy mix score and the average score of the transactions in the block. (iii) For this reason, nodes prefer including transactions with higher scores in a block, resulting in a higher probability of such transactions being included. (iv) Given any two units of gas with the same product of $score$ and $amount$, more execution time for contracts can be purchased for the unit having a higher $score$.

We disincentivize nodes from exhibiting byzantine behavior by reducing their probability to participate in block proposition and, effectively, lowering their reputation. However, we acknowledge that slashing is a severe penalty as it halves wallet score, which has severe implications on the likelihood of getting selected as a proposer. Thus, slashable offenses must be provably attributable based on an invalid block proposal. An invalid proposal consists of an invalid block number, timestamp, state hash, transaction set, or signature. Additionally, to handle crash faults while disincentivizing being offline for extended periods, nodes incur a minor penalty if they fail to propose a block when they should.

Given these incentives and disincentives, the primary objective of GreenCoin is not to be a medium of exchange, store of value, or unit of account. Instead, GreenCoin's primary objective is to act as a carbon reputation credit, where an account that possesses more coins of a higher score is perceived as more reputable by the protocol. This reputation is reflected in its privileges. As a node continues to act honestly and consume a high amount of renewable energy, its reputation and privileges increase over time. Conversely, if a node exhibits negative behaviors, GreenCoin drastically reduces its reputation by first removing coins with higher scores during slashing.

As for Sybil attacks, GreenCoin is resistant to such attacks in the permissioned context, but we are yet to validate this property in the permissionless context. Firstly, in the permissioned context, the identity of each existing and new node is known beforehand by every party in the system, making Sybil attacks virtually impossible. Secondly, the GPoS mechanism ensures that the probability of a node getting selected for block proposition is proportional to its wallet score. Thus, generating multiple accounts with a trivial wallet balance will not adversely affect the operation of the system with regard to block proposition. Lastly, the TEE requirement for all nodes prevents malicious agents from duplicating the most favorable Depot attestation for multiple nodes at geographically distinct locations.

V. Evaluation

In this section, we evaluate multiple aspects of GPoS and their impact on individual nodes in the GreenCoin environment. The prototype implementation of GreenCoin is designed for permissioned deployment in regional settings where users are not anonymized. That is, each user is authenticated into the system and only authenticated users may participate.

This use case originates with Internet of Things (IoT) applications for ecology (cf. [53]) and agriculture (cf. [54], [55]) where regional participants (who are concerned about carbon footprint) wish to use a tamper-proof ledger (with attribution) to share data, but not necessarily with an anonymized global community of users. Examples of data that these users wish to share include data on pest infestations (agriculture), predator movement (ecology), aquifer health, etc. Indeed, we

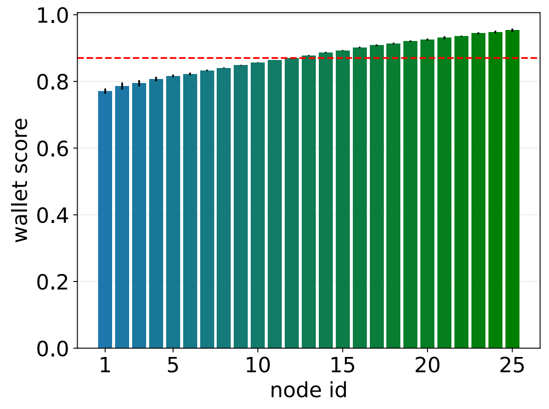


Fig. 3: Wallet scores after mining 1000 blocks. The red dashed line represents the initial wallet score. Fuel mix range increases from node 1 ($[0.00, 0.04]$) to node 25 ($[0.96, 1.00]$) at an increment of 0.04.

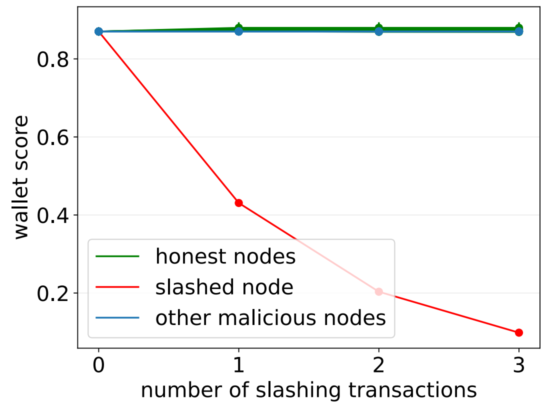


Fig. 4: Effect of slashing on wallet score of a single malicious proposer. All the nodes have the same range of fuel mix scores.

plan to use the GreenCoin prototype with our collaborators as part of ongoing field deployments of IoT applications [56]–[58]. We note, however, that GreenCoin and GPoS are not specific to these use cases; we have developed this prototype implementation for these domains to facilitate "real world" validation by and feedback from a user community.

To evaluate GreenCoin, we first study how fuel mix affects block rewards and in turn wallet score of an account. To demonstrate the impact of slashing on wallet score, we then present experimental results for running with both honest and malicious nodes. Finally, we evaluate the impact of the fuel mix score of a region on the ability of a node to execute smart contracts.

We perform our experiments using 25 virtual machine instances in a private cloud running Eucalyptus [59]. Each instance has a 2GHz CPU and 4GB of memory. Each instance is a node in a GreenCoin deployment. We assume that there are 25 accounts, each tied to a different node for the duration of the experiments. Unless otherwise specified, we perform each experiment 10 times and present the average values. We also present the standard deviation along with the mean when applicable, as error bars.

A. Block Reward

To demonstrate how the fuel mix score of a region affects block reward, we assume each of the 25 nodes is located in a region with a different fuel mix score. The node with ID 1 has the lowest range of fuel mix scores ($[0.00, 0.04]$). Each subsequent node with a higher ID has a range of fuel mix scores that is 0.04 higher than the previous node. Hence, the node with ID 25 has the highest range of fuel mix scores ($[0.96, 1.00]$). We manipulate the interface to Depot to get a random fuel mix score for each node within the appropriate range. Irrespective of its location, each account starts with a collection of 100 coins total – 10 of each score ranging from 0.1 to 1.0 at an increment of 0.1. This amounts to a wallet score of 0.8702. We run the experiment until 1000 blocks are mined, each block having a single transaction. In each of these transactions, a node sends a coin with a score of 0.5 and an amount of 1.0 to the node with the next higher ID (the last node sends the coin to the first node). We keep the score and the amount of transferred coins constant so that the final wallet score is dependent exclusively on the fuel mix of a region rather than the executed transactions.

Figure 3 shows the final wallet score of the nodes. The red dashed line represents the initial wallet score. As expected, nodes situated in regions with higher fuel mix scores experience an increase in their wallet score. The node with the highest range of fuel mix score reaches a wallet score of 0.9533, an increase of 0.0831. Conversely, nodes situated in regions with lower fuel mix scores experience a decrease in their wallet score. The node with the lowest range of fuel mix score reaches a wallet score of 0.7709, a decrease of 0.0993.

B. Account Slashing

To demonstrate the effect of slashing, we divide the nodes into two groups – 13 honest nodes and 12 malicious nodes. We manipulate the interface to Depot to get random fuel mix scores that fall within the same range for all the nodes ($[0.96, 1.00]$), so that any change in the final wallet score is only due to the slashing of nodes without any effect from disparate block rewards. All the nodes start with the same collection of coins as described in Section V-A. One of the malicious nodes proposes multiple faulty blocks. Each of the blocks contains a single transaction that transfers a coin with a score of 0.5 and an amount of 1.0 from the proposer to the next malicious node. The execution of this small number of transactions itself does not change the wallet score of the proposer significantly. All the malicious nodes vote dishonestly to declare the faulty blocks as valid and any valid block proposed by the honest nodes as invalid. We run the experiment until the wallet score of the faulty node drops below 0.2.

Figure 4 shows the wallet scores each time the faulty proposer is slashed. The red line represents the wallet score of the faulty proposer. As described in Section IV-E, each time an account is slashed its wallet score falls to half of its pre-slashed value or less. As the faulty proposer starts with

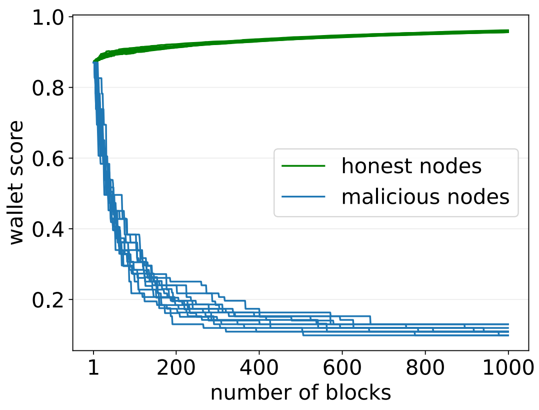


Fig. 5: Deviation of wallet scores between honest nodes and malicious nodes over a prolonged period. All the nodes have the same range of fuel mix scores.

a wallet score of 0.8702, four slashing events are enough to make the wallet score drop below 0.2. In this case, it drops to 0.0984. The wallet scores of all other nodes remain relatively unchanged.

To understand how wallet scores of honest and malicious nodes stabilize over time, we construct a second experiment with the same topology and seed account balance as the previous experiment. As in the previous experiment, all malicious nodes vote falsely. Additionally, whenever a malicious node becomes the proposer it proposes a faulty block. This behavior is different from the previous experiment, where only one specific malicious node was proposing faulty blocks and the other malicious nodes were following a more passive approach by only voting falsely. We run the experiment until 1000 blocks are generated. Figure 5 shows the deviation in wallet scores between the honest and the malicious nodes. Wallet scores of honest nodes gradually creep up to ~ 0.95 whereas those of malicious nodes fall drastically down to ~ 0.10.

C. Smart Contract Execution

As an example of smart contract execution, we use GreenCoin to compute the Pearson correlation coefficient between two time series of meteorological measurements (air temperature and humidity, in this example). For agricultural applications such as frost prevention where the activation of (possibly expensive) frost prevention measures depends on trustworthy meteorological data, we believe smart contracts will be useful.

We use the El Niño dataset (temperature and humidity) from UCI Machine Learning repository [60] for this experiment. We assume that each participant in the GreenCoin blockchain operates a set of local meteorological sensors and they use GreenCoin to publish the sensor results (taken from the dataset in this fictitious example) to the ledger via a local GreenCoin node.

As in section V-A, we start with all the nodes having a different range of fuel mix scores but the same wallet score. One node creates a contract to log temperature data, one node creates a contract to log humidity data, and another

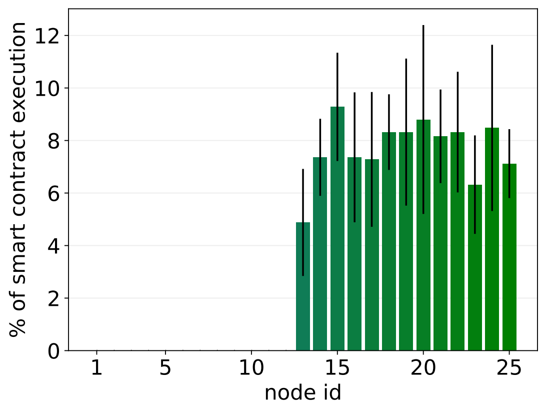


Fig. 6: Percentage of smart contract execution by each node. The execution threshold of each smart contract is set to 0.5. Nodes with ID 1 to 12 have fuel mix scores falling below 0.5. Fuel mix scores of node with ID 13 straddles 0.5 while other nodes have fuel mix scores higher than 0.5.

node creates a contract to compute the correlation between temperature and humidity. To make sure all nodes end up creating the same number and similar types of transactions, the other nodes also create such contracts, although the experiment uses only three contracts. Each node creates a transaction to log two temperature and two humidity values, resulting in 50 data points each for temperature and humidity. Finally, each node creates a transaction to compute the Pearson correlation coefficient between temperature and humidity. Therefore, there are 125 transactions executing smart contracts. Each smart contract has an execution threshold of 0.5, i.e., a blockchain node must have a minimum fuel mix score of 0.5 to propose a block containing a transaction to execute a smart contract. As GreenCoin allows execution of a smart contract by the proposer only, it means a node must have a minimum fuel mix score of 0.5 to execute a smart contract. We run the experiment until all the smart contracts have been executed. Each block contains zero or one transaction.

Figure 6 shows the percentage of smart contracts executed by each node. Nodes with fuel mix lower than the execution threshold, i.e., 0.5 fail to execute any transaction involving execution of smart contracts. All other nodes end up executing a varied percentage of transactions involving execution of smart contracts as expected.

VI. Conclusions and Future Work

Cryptocurrencies provide new functionality that improves information trustworthiness and integrity. Their energy efficiency is the subject of ongoing research and commercial development. To this end, we have investigated GreenCoin – a cryptocurrency that uses attested location to attach indelible attributes to each coin representing renewable energy usage during its creation. GreenCoin relies on GPoS, which is a new Proof-of-Stake protocol for marking each coin with a "greenness" score, determining wallet score, slashing malicious behavior, and implementing smart contracts that only execute computations when the coins that pay for them have green scores above a specified threshold.

We validate our assumptions using a permissioned prototype of GreenCoin. Our results indicate that our approach effectively implements coin scoring, incentivizes both truthful and "green" participation, and implements a smart contract with "computational accountability" with respect to the use of renewable energy in cryptocurrency systems.

As part of our future work, we plan to deploy GreenCoin in IoT application deployments as part of ongoing collaborative work. We also plan to investigate its feasibility in implementing renewable accountability for permissionless blockchains and cryptocurrencies.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21260, 2008.

[2] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[3] J. Reed, "Litecoin: An introduction to litecoin cryptocurrency and litecoin mining," 2017.

[4] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[5] "Dogecoin – an open-source peer-2-peer digital currency," https://dogecoin.com/, accessed: 17-Apr-2023.

[6] "Home — monero – secure, private, untraceable," https://www.getmonero.org/, accessed: 17-Apr-2023.

[7] "Solana — web3 infrastructure for everyone," https://solana.com/, accessed: 17-Apr-2023.

[8] "Cardano — home," https://cardano.org/, accessed: 17-Apr-2023.

[9] "Polkadot : Web3 interoperability — decentralized blockchain," https://polkadot.network/, accessed: 17-Apr-2023.

[10] "The merge is here: Ethereum has switched to proof of stake," https://www.technologyreview.com/2022/09/15/1059520/the-merge-is-here-ethereum-has-switched-to-proof-of-stake/, accessed: 17-Apr-2023.

[11] J. Ménétrey, C. Göttel, A. Khurshid, M. Pasin, P. Felber, V. Schiavoni, and S. Raza, "Attestation mechanisms for trusted execution environments demystified," in *Distributed Applications and Interoperable Systems: 22nd IFIP WG 6.1 International Conference, DAIS 2022, Held as Part of the 17th International Federated Conference on Distributed Computing Techniques, DisCoTec 2022, Lucca, Italy, June 13-17, 2022, Proceedings*. Springer, 2022, pp. 95–113.

[12] USEIA, "Us energy information administration web site," 2023, https://www.eia.gov.

[13] "The University of California, Santa Barbara, RiPiT Project," 2023, https://sites.cs.ucsb.edu/ rich/ripit.html.

[14] "The University of Chicago RiPiT Project," 2023, http://ripit.uchicago.edu.

[15] B. Sriman, S. Ganesh Kumar, and P. Shamili, "Blockchain technology: Consensus protocol proof of work and proof of stake," in *Intelligent Computing and Applications: Proceedings of ICICA 2019*. Springer, 2021, pp. 395–406.

[16] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[17] B. Cao, Z. Zhang, D. Feng, S. Zhang, L. Zhang, M. Peng, and Y. Li, "Performance analysis and comparison of pow, pos and dag based blockchains," *Digital Communications and Networks*, vol. 6, no. 4, pp. 480–485, 2020.

[18] O. Vashchuk and R. Shuwar, "Pros and cons of consensus algorithm proof of stake. difference in the network safety in proof of work and proof of stake," *Electronics and Information Technologies*, vol. 9, no. 9, pp. 106–112, 2018.

[19] A. De Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, no. 5, pp. 801–805, 2018.

[20] V. Kohli, S. Chakravarty, V. Chamola, K. S. Sangwan, and S. Zeadally, "An analysis of energy consumption and carbon footprints of cryptocurrencies and possible solutions," *Digital Communications and Networks*, vol. 9, no. 1, pp. 79–89, 2023.

[21] Z. Wang, Q. Li, J. Song, H. Wang, and L. Sun, "Towards ip-based geolocation via fine-grained and stable webcam landmarks," in *Proceedings of The Web Conference 2020*, 2020, pp. 1422–1432.

[22] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, where's that ip? circumventing measurement-based ip geolocation," in *Proceedings of the 19th USENIX conference on Security*, 2010, pp. 16–16.

[23] P. Callejo, M. Gramaglia, R. Cuevas, and A. Cuevas, "A deep dive into the accuracy of ip geolocation databases and its impact on online advertising," *IEEE Transactions on Mobile Computing*, 2022.

[24] "Ip geolocation and online fraud prevention — maxmind," https://www.maxmind.com/en/home, accessed: 15-Apr-2023.

[25] "American registry for internet numbers," https://www.arin.net/, accessed: 15-Apr-2023.

[26] "Ripe network coordination center," https://www.ripe.net/, accessed: 15-Apr-2023.

[27] M. Cozar, D. Rodriguez, J. M. Del Alamo, and D. Guaman, "Reliability of ip geolocation services for assessing the compliance of international data transfers," in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2022, pp. 181–185.

[28] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards ip geolocation using delay and topology measurements," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 71–84.

[29] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A learning-based approach for ip geolocation," in *Passive and Active Measurement: 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings 11*. Springer, 2010, pp. 171–180.

[30] J. Saxon and N. Feamster, "Gps-based geolocation of consumer ip addresses," in *Passive and Active Measurement: 23rd International Conference, PAM 2022, Virtual Event, March 28–30, 2022, Proceedings*. Springer, 2022, pp. 122–151.

[31] A. Vaish, A. Kushwaha, R. Das, and C. Sharma, "Data location verification in cloud computing," *International Journal of Computer Applications*, vol. 68, no. 12, 2013.

[32] A. Noman and C. Adams, "Hardware-based dlas: Achieving geolocation guarantees for cloud data using tpm and provable data possession," in *2014 17th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2014, pp. 280–285.

[33] S. Park, J. N. Yoon, C. Kang, K. H. Kim, and T. Han, "Tgvisor: A tiny hypervisor-based trusted geolocation framework for mobile cloud clients," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, 2015, pp. 99–108.

[34] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/Ispa*, vol. 1. IEEE, 2015, pp. 57–64.

[35] L. ARM, "Arm security technology-building a secure system using trustzone technology," PRD-GENC-C. ARM Ltd. Apr.(cit. on p.), Tech. Rep., 2009.

[36] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." *Hasp@ isca*, vol. 10, no. 1, 2013.

[37] M. Crone, "Towards attack-tolerant trusted execution environments: Secure remote attestation in the presence of side channels," 2021.

[38] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy execution on mobile devices: What security properties can my mobile platform give me?" in *Trust and Trustworthy Computing: 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings 5*. Springer, 2012, pp. 159–178.

[39] H. Vill, "Sgx attestation process," 2017.

[40] S. Weiser and M. Werner, "Sgxio: Generic trusted i/o path for intel sgx," in *Proceedings of the seventh ACM on conference on data and application security and privacy*, 2017, pp. 261–268.

[41] J. Thalheim, H. Unnibhavi, C. Priebe, P. Bhatotia, and P. Pietzuch, "Rkt-io: A direct i/o stack for shielded execution," in *Proceedings of the Sixteenth European Conference on Computer Systems*, 2021, pp. 490–506.

[42] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted tee systems," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1416–1432.

[43] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on intel sgx," *arXiv preprint arXiv:2006.13598*, 2020.

[44] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings*. Springer, 2006, pp. 1–20.

[45] Y. Yarom and K. Falkner, "Flush+ reload: A high resolution, low noise, l3 cache side-channel attack," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 719–732.

[46] C. Canella, J. Van Bulck, M. Schwarz, M. Lipp, B. Von Berg, P. Ortner, F. Piessens, D. Evtyushkin, and D. Gruss, "A systematic evaluation of transient execution attacks and defenses." in *USENIX Security Symposium*, 2019, pp. 249–266.

[47] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 142–157.

[48] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *USENIX Security Symposium (SEC)*, 2018.

[49] X. Xin, "Titan m makes pixel 3 our most secure phone yet," *Google (Oct. 2018). url: https://www. blog. google/products/pixel/titan-m-makespixel-3-our-most-secure-phone-yet*, 2018.

[50] "energy mix," https://archive.unescwa.org/energy-mix, accessed: 14-Apr-2023.

[51] "California iso – supply, today's outlook," https://www.caiso.com/todaysoutlook/Pages/supply.aspx, accessed: 14-Apr-2023.

[52] K. Mammadzada, M. Iqbal, F. Milani, L. García-Bañuelos, and R. Matulevičius, "Blockchain oracles: a framework for blockchain-based applications," in *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18*. Springer, 2020, pp. 19–34.

[53] A. R. Elias, N. Golubovic, C. Krintz, and R. Wolski, "Where's the Bear? – Automating Wildlife Image Processing Using IoT and Edge Cloud Systems," in *Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on*. IEEE, 2017, pp. 247–258.

[54] N. Golubovic, R. Wolski, C. Krintz, and M. Mock, "Improving the Accuracy of Outdoor Temperature Prediction by IoT Devices," in *IEEE Conference on IoT*, 2019.

[55] N. Golubovic, A. Gill, C. Krintz, and R. Wolski, "CENTAURUS: A Cloud Service for K-means Clustering," in *IEEE DataCom*, Nov. 2017.

[56] C. Krintz, R. Wolski, N. Golubovic, B. Lampel, V. Kulkarni, B. Sethuramasamyraja, B. Roberts, and B. Liu, "SmartFarm: Improving Agriculture Sustainability Using Modern Information Technology," in *KDD Workshop on Data Science for Food, Energy, and Water*, Aug. 2016.

[57] "UCSB SmartFarm," https://sites.cs.ucsb.edu/ ckrintz/projects/, [Online; accessed 17-April-2023].

[58] "UCSB Edible Campus," https://sustainability.ucsb.edu/ediblecampus, [Online; accessed 17-April-2023].

[59] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[60] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml