2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 979-8-3315-0934-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/CCGRID64434.2025.00041

COoL-TEE: Client-TEE Collaboration for Resilient Distributed Search

Matthieu Bettinger
LIRIS-DRIM INSA Lyon
Lyon, France
matthieu.bettinger@insa-lyon.fr

Etienne Rivière
ICTEAM UCLouvain
Louvain-la-Neuve, Belgium
etienne.riviere@uclouvain.be

Sonia Ben Mokhtar
LIRIS-DRIM CNRS
Lyon, France
sonia.ben-mokhtar@cnrs.fr

Anthony Simonet-Boulogne
iExec Blockchain Tech
Lyon, France
anthony.simonet-boulogne@iex.ec

***Abstract*—Current marketplaces rely on search mechanisms with distributed systems but centralized governance, making them vulnerable to attacks, failures, censorship and biases. While search mechanisms with more decentralized governance (e.g., DeSearch) have been recently proposed, these are still exposed to *information head-start attacks* (IHS) despite the use of Trusted Execution Environments (TEEs). These attacks allow malicious users to gain a head-start over other users for the discovery of new assets in the market, which give them an unfair advantage in asset acquisition. We propose COoL-TEE, a TEE-based provider selection mechanism for distributed search, running in single- or multi-datacenter environments, that is resilient to information head-start attacks. COoL-TEE relies on a Client-TEE collaboration, which enables clients to distinguish between slow providers and malicious ones. Performance evaluations in single- and multi-datacenter environments show that, using COoL-TEE, malicious users respectively gain only up to 2% and 7% of assets more than without IHS, while they can claim 20% or more on top of their fair share in the same conditions with DeSearch.**

***Index Terms*—distributed search, marketplace, Trusted Execution Environments (TEE).**

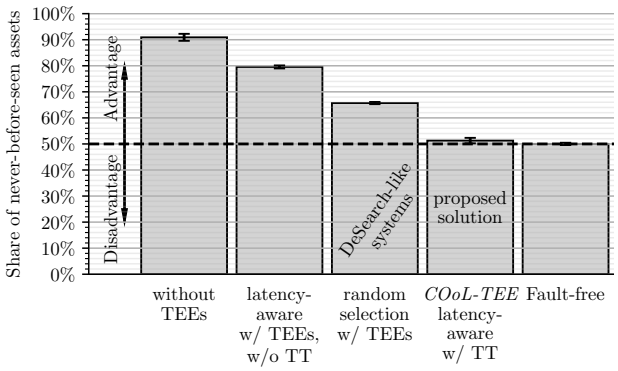


Fig. 1: Share of never-before-seen assets discovered first by malicious consumers ($\frac{50}{100}$ of total consumers), under different configurations. $\frac{4}{8}$ search providers are malicious and the total system load is 75%. *The closer the values are to 50%, the better, indicating lower malicious IHS advantage.*

I. Introduction

Centralized e-commerce platforms (e.g., Amazon or Alibaba) are vulnerable to attacks, failures, and biases [1]–[6]. Recent solutions working towards more decentralized governance in search systems [7]–[9] include DeSearch [7], which uses Trusted Execution Environments (TEEs) to protect against censorship and bias. DeSearch relies on users contributing computing resources to run the search protocol. These users deploy the DeSearch pipeline on their own TEE-enabled servers or on cloud virtual machines, where TEEs are now widely available (e.g., for Intel SGX, Alibaba, Azure, IBM Cloud, OVH). These servers crawl the market and maintain an index, on top of which they provide a search service to users. TEEs at the protocol-level protect the correctness and confidentiality of the search mechanism, while the decentralized ownership mitigates attacks from single owners of VMs.

In a context where assets in a marketplace are valuable, scarce, and often both, it is important to ensure that search service providers cannot favor some users and penalize others. However, the last decade has shown that attacks on marketplaces are commonplace, as soon as there is value to be earned [10], [11]. In the context of search, malicious search providers can delay or restrict access to knowledge about new assets for some users, thereby giving a head-start to others, which is comparable to forms of insider trading in financial markets. We call this behavior an *information head-start* (IHS) attack. We measure the impact of IHS attacks by the share of new assets discovered by malicious consumers before honest consumers. We show in this paper that recent work on decentralized search mechanisms for marketplaces is vulnerable to IHS attacks: despite the use of TEEs, malicious service providers have the power to delay responses to honest consumers, in favor of malicious consumers, which we highlight as a novel attack in the context of TEEs. Colluding malicious consumers can accentuate the providers' attack by loading honest providers with requests (without a Denial-of-Service attack), increasing those providers' end-to-end latency, so as to steer honest, latency-aware consumers towards malicious providers who will then attack them.

We introduce COoL-TEE, which combines Client-side Optimization of Latencies with provider-side TEEs to protect against IHS attacks. COoL-TEE provides a client-side

This work was supported by a French government grant managed by the Agence Nationale de la Recherche under the France 2030 program, reference "ANR-22-PEFT-0002" as well as the ANR Labcom program, reference "ANR-21-LCV1-0012".

2993-2114/25/$31.00 ©2025 IEEE
DOI 10.1109/CCGRID64434.2025.00041

Provider Selection Module that selects providers based on past latencies and on trustworthy measurements by TEEs, allowing consumers to avoid high-latency or malicious providers. Figure 1 illustrates the impact of IHS and its mitigation using COoL-TEE in a single datacenter: in an unprotected system (1st bar) malicious providers and malicious clients can gain a significant advantage, i.e., discover many new assets before others, due to the absence of integrity and confidentiality guarantees. Simply adding TEEs and choosing low-latency providers (2nd bar) is not enough: TEEs like Intel SGX do not have access to a Trustworthy Time (TT) source, so malicious providers can manipulate their notion of time and disguise themselves as well-performing providers. DeSearch's random selection (3rd bar) side-steps being manipulated by malicious providers (by ignoring such misinformation attempts), but is still vulnerable to IHS through delayed responses from randomly selected malicious providers. COoL-TEE (4th bar) incorporates a Trusted Time mechanism and mitigates IHS attacks. Indeed, COoL-TEE is the closest to the behavior of a fault-free system (5th bar). While recent work on TEE Trusted Time [12], [13] propose resilience inside a TEE execution against a malicious host adding delays, we contribute a solution that extends resilience to the protocol level, during interactions between remote users and TEEs.

We evaluate COoL-TEE in both cloud and simulated environments (source code available [14]), comparing its performance against decentralized search and provider selection systems [7], [15]–[17], under various configurations and attacker strengths. In particular, we evaluate how much consumers gain or lose in timely market information through IHS, i.e., in opportunities to acquire new assets. We present results where up to all providers may be malicious, with 100 consumers and 8 providers in a single datacenter, as well as with 1000 consumers and 48 providers in multiple geo-distributed datacenters. We show that, with COoL-TEE, malicious consumers do not get an edge over others in terms of fresher market state information, as long as there is enough honest provider throughput to serve all honest requests, i.e., less than 2% and 7% advantage respectively in single- and multi-datacenter experiments, including Figure 1.

The paper is structured as follows: Section II presents the ecosystem and threat models. Section III describes our proposed solution, COoL-TEE. Sections IV and V cover our experimental protocol and results. Finally, Section VI discusses related work and Section VII concludes the paper.

II. Model & problem statement

We first detail the marketplace search model then formalize our system and attacker models.

A. Marketplace search model

The system consists of three types of entities, illustrated in Figure 2: (i) the *market* (right) is the (distributed) storage system where assets are registered. (ii) *Consumers* (left) query the search mechanism; and (iii) *Providers* (middle) provide

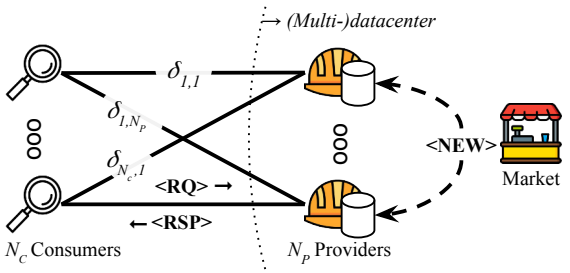


Fig. 2: *Marketplace search model* Consumers send requests to Providers about the market's state; Providers respond with a list of assets they learned from the Market.

this search mechanism, updating an index of the market state to respond to consumers' requests.

All consumers aim to be the first to discover new assets that appear on the marketplace. A subset of consumers tries to improve its asset-discovery chances by colluding with a subset of providers, e.g., in exchange for bribes. Those subsets of consumers and providers are considered malicious.

We call assets not yet discovered by any consumer *Never-Before-Seen assets* (NBSa). When they are first discovered by a consumer, they become a *discovered NBSa* (dNBSa) by this consumer. We call attacks impacting a consumer's share of dNBSa *information head-start* (IHS) attacks. As shown in Figure 2, we consider N_C consumers and N_P providers on a network. Providers are servers hosted in a (multi-)datacenter environment on TEE-enabled machines, while consumers are clients either on the same cloud infrastructure or on the Internet. The network delay between consumer i and provider j is $\delta_{i,j}$. Two protocols run in parallel: the market and the search mechanism. New assets are broadcast to all providers at rate λ_a as **NEW** messages. Consumers send requests **RQ** to providers at rate λ_r. Providers handle requests with a FIFO policy, with a queue delay δ_q and a fixed response time D. Responses **RSP** contain assets matching the request's keywords. We assume consumers only use this search mechanism to discover assets, i.e., they do not crawl the market themselves, due to computational constraints. The index served by a provider is assumed to be updated synchronously with respect to others providers, e.g., periodically or, if the market is hosted on a blockchain, when a new block appears.

B. Trusted Execution Environments (TEEs) and DeSearch [7]

Intel Software Guard eXtensions [18] (Intel SGX) is a hardware-based Trusted Execution Environment that provides confidentiality and integrity guarantees for code and data running inside that TEE. DeSearch [7] is a decentralized search protocol using Intel SGX TEEs against censorship and bias attacks. DeSearch follows a 3-step pipeline: crawling assets, indexing them, and serving search requests. Our work focuses on the search step. Using TEEs, DeSearch guarantees that the index is verifiably correct and complete. TEEs also ensure search responses are complete and tamper-proof, while Circuit-ORAM [19] hides memory access patterns. Due to

Circuit-ORAM's memory-access obfuscation, concurrent access on the same index is not possible. Therefore, in DeSearch and in our model, TEEs serve requests sequentially.

C. Threat model

Intel SGX augmented with oblivious RAM protects the integrity and confidentiality of the search mechanism. However, providers that host the TEE retain control over the machine surrounding the TEE enclave: malicious providers can delay responses to honest consumers even in the presence of TEEs. We identify this attack vector as a lever for IHS attacks.

We assume proportions c_M of consumers and p_M of providers are malicious. We assume honest and malicious actors have similar computing power, proportional to (1-p_M) and p_M for providers, and similarly for consumers with respect to c_M. This computing power corresponds to maximum request emission and handling throughputs respectively for consumers and providers. All consumers compete over the same stream of assets: preventing honest consumers from discovering NBSa first is a valid strategy for malicious actors.

We now present two provider-side attacks and a consumer-side attack on the marketplace search model and compare them to the fault-free case (Figure 3a), where all providers serve all consumers with timely and correct results.

1) Content attacks: Without TEEs, malicious providers can read requests and send irrelevant, outdated or no responses at all to honest consumers (Figure 3b). This leads to *content attacks*, which comparatively grant more asset-discovery opportunities to malicious consumers.

2) Delay attacks: Introducing TEEs protects the protocol's integrity and confidentiality but cannot prevent response delays. Indeed, malicious providers, who control the machine around the TEE, can delay responses to honest consumers by δ_{att} (Figure 3c), leading to *delay attacks*. Responses are correct but delayed, reducing freshness, i.e., received responses represent an older market state than in the fault-free case.

3) Cuckoo attacks: In *cuckoo attacks*, malicious consumers load honest providers with requests, enticing latency-aware honest consumers to switch to malicious providers. In turn, these malicious providers can perform a delay attack, resulting in a joint attack which we call a *cuckoo-delay* attack (cuckoo-D). For comparison purposes, we also evaluate *cuckoo-content attacks* (cuckoo-C), i.e., the case where providers do not operate TEEs and are so able to perform content attacks.

D. Trusted Time in Trusted Execution Environments

In TEE-based environments, malicious providers add delays in the protocol for honest consumers to reduce their dNBSa share. Indeed, without trusted time sources for TEEs, consumers can only trustlessly measure round-trip latencies by themselves. As a lever for cuckoo-delay attacks, this enables malicious providers to disguise themselves as loaded providers, among the actually-loaded honest providers, victims of the cuckoo-attack's malicious consumer request load. Already a source of trust inside the potentially malicious host, one would want to also use the TEE to measure delays in the request pipeline and inform consumers. However, Intel SGX's trusted time primitive [20] is no longer available since 2020 and other time sources are mediated by the untrusted OS [21]. Recent work [12], [13] tackle restoring a resilient wall clock to Intel SGX TEEs. Based on DeSearch's per-TEE request throughput in the hundreds per second, we require a fine-grained wall-clock time source in milliseconds that is resilient to TEE-level delay attacks (i.e., a T2 timer in Alder et al.'s TEE Trusted Timer categorization [21].) We therefore leverage Triad [13], a multi-TEE protocol that avoids disruptions in the TEE's notion of elapsed time through awareness of TEE enclave exits (using AEX-Notify [22]), cooperation with TEEs in the same datacenter, and a remote time authority for setup and in cases where all TEEs simultaneously lose their notion of time continuity. Note that such trusted time mechanisms protect against delay attacks inside the TEE, not at the protocol-level, which is the focus of this paper. Delay attacks on TEEs themselves [21] operate in a similar way to ours (see Figure 3c), the TEE being the consumer and the Trusted Time source the trusted component surrounded by a potential attacker. With a resilient time source, the TEE can now measure elapsed time for operations occurring inside it and report those measurements to consumers.

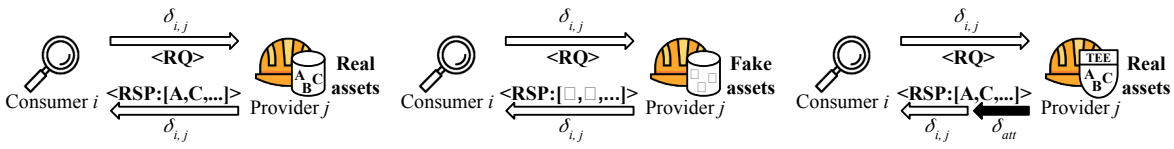
III. COoL — Client-side Optimization of Latencies

A consumer wants the freshest information about the market, that is, responses that contain the newest relevant assets. To that end, COoL-TEE enables honest consumers to preferentially select providers that serve responses with lower latencies than others, i.e., that are closer, faster, and honest. Trustlessly with respect to other consumers and providers hosting the TEEs, COoL-TEE selects providers based on past latencies the consumer personally experienced and on request residence time measurements from TEEs.

To present our solution, we first focus on the timing aspects of the two parallel protocols, as shown in Figure 4: market indexing (right) and our proposed provider-selection and search mechanisms (left). We then detail the *Provider Selection Module* (PSM) that performs Client-side Optimization of Latencies, leveraging TEE Trusted Time.

A. Protocol delays and timing

Figure 4 shows the behavior of a consumer i using the search mechanism and providers 1 to N_P: (1) At t_{gen}, consumer i generates a request $\mathbf{RQ}$. (2) The PSM directs the request to provider j at t_{send} after a delay δ_{PSM}. (3) The request is sent to provider j with a one-way network delay $\delta_{i,j}$. (4) The request waits in the TEE's queue for δ_{queue} until service starts at t_{serv}, with a service time $\delta_{serv} = D$. The TEE measures the request's waiting time δ_{queue} and reports it in the response. (5) After service, the TEE sends the response $\mathbf{RSP}$, which may be delayed by δ_{att} if the host is malicious. The response is then sent back to consumer i with a delay $\delta_{i,j}$. (6) At t_{recv}, consumer i receives the response and logs the round-trip latency minus the queue waiting time as $\Delta^k_{i,j}$, the k^{th} measurement with respect to provider j.



(a) Fault-free (b) Content attack (c) Delay attack

Fig. 3: Provider-side attack models on marketplace search

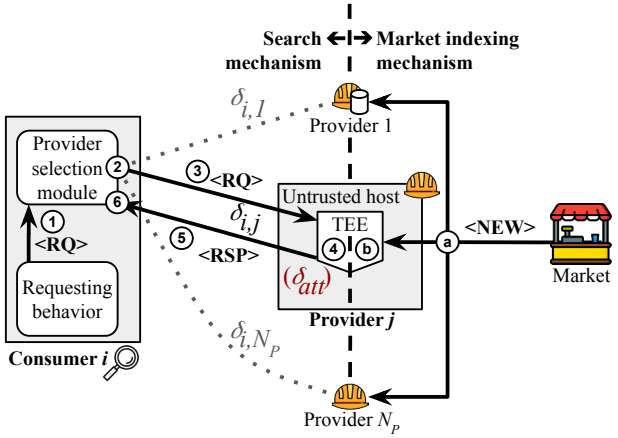


Fig. 4: High-level system architecture

Concurrently, new assets are broadcast to all providers (step a) and indexed at t_{index} (step b). For an asset A to appear in a response, $t_{index} < t_{serv}$. The freshness of a response is $\delta_{fresh} = D + \delta_{i,j} + \delta_{att}$, i.e., the delay from when the TEE starts serving a request to when the consumer receives the response. Honest consumers maximize freshness by choosing close, fast, and honest providers. As it is difficult to trustlessly measure the individual delays that make up δ_{fresh}, notably the return-trip outside the TEE $\delta_{i,j} + \delta_{att}$ (consumer and TEE clock phases are not assumed to be synchronized), honest consumers rely with COoL-TEE on minimizing round-trip latencies substracted by TEE-measured trustworthy delays. To appropriately substract the TEE-measured queue waiting time from the consumer-measured round-trip latency, as part of the communication setup with each TEE, e.g., after TEE attestation, a consumer measures the drift rate with the TEE's time source. To that end, subprotocols of clock synchronization protocols like NTPsec [23] can be used, e.g., in that case, the clock filter and discipline algorithms [24].

B. COoL provider selection

A key mechanism in COoL-TEE is the Provider Selection Module (PSM). Consumers want the freshest information about the market's state. To minimize average response latencies, each consumer i's PSM uses *Client-side Optimization of Latencies* (COoL) by aggregating past round-trip latencies and TEE delay measurements as $\Delta^k_{i,j}$ and then adjusting the probability of selecting each provider. Consumer i then directs a proportion $r_{i,j}$ of their total request throughput λ^i_r towards each provider j. In its implementation [14], COoL-TEE uses a sliding-window average of the s latest latencies from each provider to compute the average latencies.

Algorithm 1: PSM.UpdateSelectionRatios

Input: $\{r^{in}_j\}$ set of initial provider selection ratios; $\{\Delta_j\}$ set of per-provider observed latencies sets
Data: x exploration coefficient; K_p, K_d PD-controller coefficients; s sliding window size; c cluster threshold; a attrition ratio
Output: $\{r^{out}_j\}$ set of updated provider selection ratios

1 **begin**
2 **foreach** $j \in [\![1; N_P]\!]$ **do**
3 $\Delta^{avg}_j \leftarrow \frac{1}{s}\sum_{k=1}^{s} \Delta^k_j$;
4 $\Delta^{prev,avg}_j \leftarrow \frac{1}{s}\sum_{k=s+1}^{2s} \Delta^k_j$;
/* Cluster best providers in B_{prov} and worst providers in W_{prov} */
5 $\Delta^{avg}_{best} \leftarrow \min_{j\in[\![1;N_P]\!]}(\Delta^{avg}_j)$;
6 $B_{prov} \leftarrow \{j \in [\![1; N_P]\!] \mid \Delta^{avg}_j \leq \Delta^{avg}_{best} + c\}$;
7 $W_{prov} \leftarrow \{j \in [\![1; N_P]\!] \mid \Delta^{avg}_j > \Delta^{avg}_{best} + c\}$;
8 $\Delta^{avg}_{target} \leftarrow \frac{1}{|B_{prov}|}\sum_{j\in B_{prov}} \Delta^{avg}_j$;
/* PD-control best providers */
9 **foreach** $j \in B_{prov}$ **do**
10 $e_j \leftarrow \frac{\Delta^{avg}_{target} - \Delta^{avg}_j}{\Delta^{avg}_{target}}$;
11 $d_j \leftarrow \Delta^{avg}_j - \Delta^{prev,avg}_j$;
12 $r^{tmp}_j \leftarrow clamp(r^{in}_j + K_p e_j + K_d d_j, 0, 1)$;
13 $S_{best} \leftarrow \sum_{j\in B_{prov}} r^{tmp}_j$;
/* Lessen use of worst providers */
14 **foreach** $j \in W_{prov}$ **do**
15 $r^{norm}_j \leftarrow (1-a) r^{in}_j$;
16 $S_{worst} \leftarrow \sum_{j\in W_{prov}} r^{tmp}_j$;
17 **foreach** $j \in B_{prov}$ **do**
18 $r^{norm}_j \leftarrow (1 - S_{worst}) r^{tmp}_j / S_{best}$;
19 **foreach** $j \in [\![1; N_P]\!]$ **do**
20 $r^{out}_j \leftarrow (1-x) r^{norm}_j + x\frac{1}{N_P}$;
21 **return** $\{r^{out}_j\}$

The PSM operates in two phases: selecting a provider for each request and updating selection ratios. In Figure 4, at step 2 of the search mechanism, the PSM selects a provider j with probability r_j. Periodically, the PSM uses Algorithm 1 to update $\{r_j\}$ based on observed latencies $\{\Delta_j\}$. In lines 3 and 4, average per-provider latency in the current and

previous windows are computed respectively as Δ_j^{avg} and $\Delta_j^{prev,avg}$. Based on the provider with the lowest average latency Δ_{best}^{avg} (line 5), providers are divided in two groups: the "best providers", whose average latency is at most $\Delta_{best}^{avg} + c$, in B_{prov}; while the rest constitute the "worst providers", in W_{prov}. Lines 9 to 12 implement a Proportional-Derivative (PD) controller on selection ratios for the best providers. The PD-controller's setpoint is the current average latency Δ_{target}^{avg} among providers in B_{prov} (line 8). The provider-specific relative error e_j to the target (line 10) as well as its derivative d_j (line 11) are computed to adjust the selection. The result of the Proportional Derivative control expression $r_j^{in} + K_p e_j + K_d d_j$ is then mapped to its closest value in the $[0, 1]$ interval (line 12). If a provider's latency Δ_j^{avg} is higher than the average Δ_{target}^{avg}, r_j decreases; if it is lower, r_j increases. Meanwhile, the worst providers lose a portion a of their selection ratios (line 15) and this loss is transferred to the best providers during a normalization step (line 18). As a last common step for all providers, selection ratios are mixed with a uniform probability (line 20): the PSM allocates a proportion x of requests to explore the latency Quality-of-Service at available providers. Indeed, we remark that this client-side provider selection can be described as a Multi-Armed Bandit problem [25]: a consumer sends a request to a chosen provider j (i.e., pulls lever j), then experiences a round-trip latency and a TEE-reported queue waiting-time (i.e., the reward). The consumer's objective is to maximize its reward: select providers in order to minimize average latencies. As is common in Multi-armed bandit settings [25], the selection policy should allocate a small fraction of requests to explore the latency-performance of each provider, to avoid exploiting sub-optimal providers asymptotically.

Algorithm 1 requires calibration of the coefficients K_p and K_d, as well as of the exploration coefficient x, so as to offer theoretical guarantees. This can be online, during the system's operation, with parameter-tuning algorithms [26], [27]. Algorithms from the Multi-Armed Bandit literature that tolerate evolving reward distributions, like Exp3 [28], may also be used. As our focus in this paper is not on convergence itself, but on the asymptotic behavior of the system where selection ratios have converged, we tune parameters manually. We obtain a system where the selection ratios converge to the valuations yielding minimal latencies in micro-benchmark scenarios, i.e., in a set of stereotypical scenarios (e.g., high- versus low-latency providers; high- versus low-throughput providers), and converge with random topologies drawn from the dataset which will be presented in the next section.

IV. Experimental protocol

This section presents how we evaluate COoL-TEE.

1) Metrics: We use discovered Never-Before-Seen assets (dNBSa) as well as the system's latency and throughput as metrics to evaluate COoL-TEE with respect to related work. The main metric is the number of dNBSa discovered by malicious consumers. If the proportion of dNBSa by malicious consumers matches their proportion of sent requests among all the whole consumer population, there is no IHS advantage. Higher or lower proportions indicate IHS advantage or disadvantage, respectively. Secondary metrics are latency and peak throughput for consumer requests and provider responses, impacting response freshness and Quality-of-Service (QoS).

2) Deployment and simulation setups: We implement COoL-TEE in C++, using DeSearch [7] as the backend. We deploy the system on Ubuntu 20.04 Azure VMs, where providers run on a 4-core SGX-enabled VM (*DC4s_v2*), and consumers on a 16-core non-SGX VM (*B16als_v2*). Both machines are in the same datacenter, with an average round-trip latency of 1.1ms (0.5ms of standard deviation in 100 pings) between them. SGX's AEX-Notify [22], required by Triad's trusted time mechanism [13], is only available for Linux kernels 6.2+, but DeSearch [7] has dependencies with Ubuntu 20.x. Therefore, we evaluate the impact of Trusted Time on a separate 32-core SGX2-enabled server and simulate DeSearch in that case as a black box with the same interface. We use these deployments to measure overheads incurred by our protocol in a distributed setting and to calibrate parameters of a simulated version of the system. Indeed, due to the high number of configurations to evaluate, we also implement a simulated version of COoL-TEE and of baselines using Omnet++ [29]: simulation experiment design points presented in this paper represent around 40k protocol runs and 1TB of requests' lifecycle traces. The code and analysis scripts are publicly available [14]. Absent in the original Triad paper [13], we also contribute a public implementation of Triad's trusted time protocol, that we integrate to COoL-TEE [14].

3) Network topologies: We consider two network topologies: (1) "same-datacenter" topology, where latencies are the same between all actors ($\delta_{net} = 0$) and (2) "multi-datacenter" topology, where latencies between consumers and providers are heterogeneous and assigned using the CloudEdge latency dataset [30]. This dataset contains the median roundtrip latencies between 69 datacenters from major cloud service providers (CSPs) and 8456 probes from RIPE Atlas [31], an Internet connectivity monitoring service, all positioned across the globe. In simulation-based experiments, our search providers are assigned to datacenter nodes in that dataset, while consumers are a subset of RIPE Atlas probes. To still take advantage of the datacenter geodistribution by CSPs among a smaller subset of datacenters, without loss of generality, we use IBM datacenters from the dataset (22 datacenters, the most numerous compared to other represented cloud service providers). We position 2 provider nodes per datacenter, except in South America (4 nodes) and in Australia (3 nodes) to account for the low number of existing IBM datacenters there (resp. 1 and 2). Additionally, we sample equal numbers of consumer nodes whose closest datacenter is in the same geographic region (i.e., 200 consumers each, in North America, South America, Europe, Asia, and Oceania.) Using that dataset, the latency distribution for users to their closest datacenter depends on the geographic region: the median latency is around 15ms to 30ms for all regions except South America (40ms). Depending on the geographic region,

80 to 95% of users access their closest datacenter with less than 50ms round-trip latency (60% for South America,) while 98% of users have less than 150ms round-trip latency.

4) Simulation parameters: Simulations involve $N_C = 100$ consumers and $N_P = 8$ providers for same-datacenter experiments and $N_C = 1000$ consumers and $N_P = 48$ providers for multi-datacenter experiments. Each provider handles up to 160 requests per second (6.25 ms/request), a throughput obtained empirically with DeSearch. Consumers send requests at rates λ_r so as to ensure a total average system load $\rho = 75\%$ (unless otherwise specified). On the market-side, assets arrive at an average $\lambda_a = 100$ assets per second. Regarding event distributions, figures present results for consumer requests emission and asset arrivals following Poisson processes, with respective rates λ_r and λ_a. We do not illustrate results in the paper for other distributions due to lack of space: with periodic events (whether requests or assets), results are similar on average to those presented, but with higher standard deviations.

5) Provider selection policies: Consumers select providers using "DeSearch-like" random selection or COoL-TEE's "COoL" selection optimizing for latency. We assess "COoL" selection both without and with Trusted Time (TT). We also separately evaluate COoL selection augmented with state-of-the-art Power-of-Two (PoT) load balancing [15], [32], called "COoL-PoT". In COoL-PoT, consumers select two providers, and the one with the shorter queue handles the request. Finally, we consider multiprovider selection policies where honest consumers send requests to k providers to probabilistically avoid sending requests only to malicious providers.

6) Attack configurations: The malicious providers' proportion p_M varies between 0 and 1. Meanwhile, in illustrated results, malicious consumers' proportion c_M in the population is set to 50% for more intuitive result description, but we discuss the influence of c_M on IHS. We implement provider- and consumer-side attack strategies defined in Section II. Delay attacks use a delay $\delta_{att} = 50ms$. Additionally, for the specific case of selection policies using Power-of-Two, we consider the attack vector that opens using that mechanism, which we call the "queue attack". In a queue attack, malicious providers keep a secondary queue outside the TEE, so that the TEE's queue always remains at a length $L \leq 1$ (considering a request being handled remains at the front of the queue until it is served). This means that from the point of view of honest providers, malicious providers more often have shorter queues than them and, consequently, should yield service of a request to them more often.

V. Results

We evaluate COoL-TEE's performance in terms of IHS advantages and latency-throughput to answer:

RQ1: What is COoL-TEE's performance in a fault-free case?

RQ2: What is its performance in an adversarial same-datacenter setting?

RQ3: What is its performance in an adversarial multi-datacenter setting?

A. Fault-free setup

To address *RQ1*, we first evaluate the impact of solution building blocks in a fault-free setting deployed on the cloud.

1) Provider selection and security mechanism overheads: Figure 5a shows overheads for provider selection policies and security features in a same-datacenter environment. Latency is measured from request generation to response reception, including provider selection time. The choice of selection policy has neglible impact on maximum throughput and average latency. Compared to an execution in SGX simulation mode ("SIM" in Figure 5a), SGX hardware mode without Circuit-ORAM ("HW") has a low latency overhead ($< 2ms$) and reduces throughput by around 11%. Compared to standalone SGX hardware mode, Circuit-ORAM increases latency by 4 to 10ms and reduces throughput by 30%.

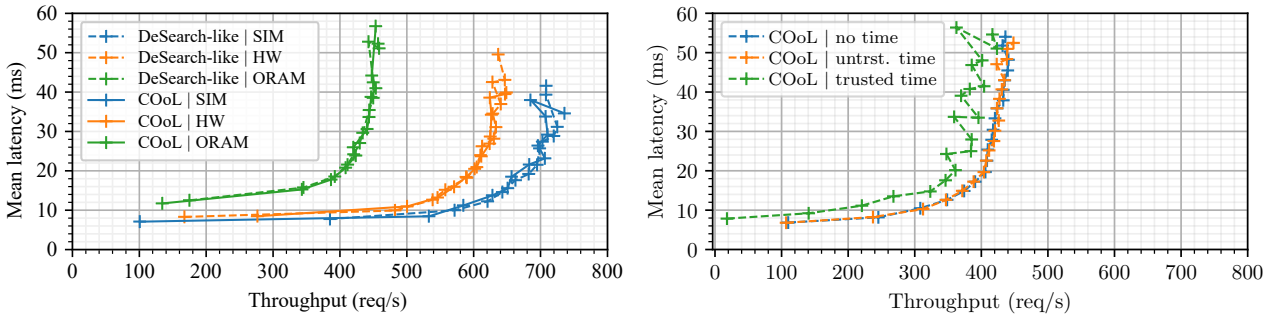
2) Time source overheads: Following the provider selection and SGX overhead analysis, we evaluate in Figure 5b the impact of the time source type on COoL-TEE's performance. Without measuring time and when using the untrusted OS time, i.e., without deploying the Triad [13] Trusted Time mechanism alongside COoL-TEE, we obtain a total peak throughput of around 440k requests per second for $N_P = 3$ providers. Using Trusted Time, the throughput is reduced by around 10 to 15%, while the latency increases by 2 to 4ms.

B. Adversarial same-datacenter setup

We now simulate attacks in a same-datacenter setup with varying proportions of malicious providers p_M, evaluating the impact of provider selection policies, of load balancing policies, of attack strategies, and of scaling out providers. We first evaluate systems without Trusted Time in Sections V-B1 to V-B4, then with Trusted Time in Section V-B5.

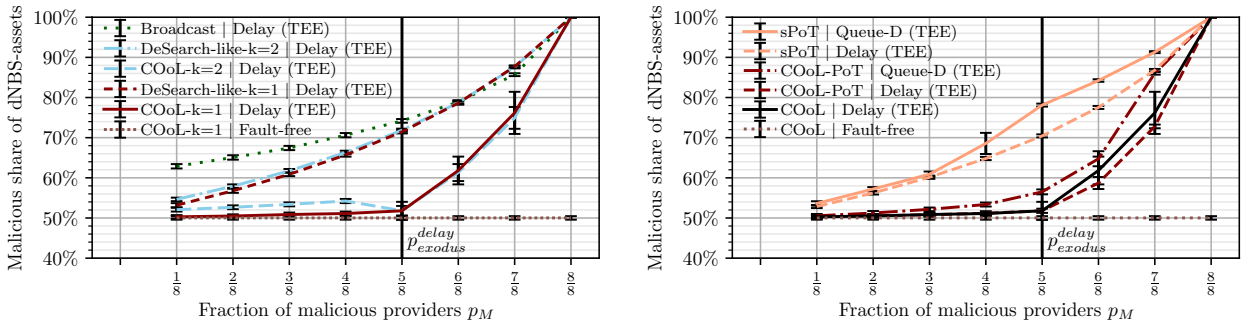
1) Multiprovider selection: We compare multiprovider selection with DeSearch-like or COoL policies, while varying $k \in \{1, 2, 8\}$, the number of selected providers per request. In the 8-provider setup, $k = 8$ corresponds to the broadcast policy used in related work [16], [17]. Figure 6a shows the information head-start (IHS) advantage for malicious consumers, measured by their share of dNBSa, under *delay attacks* and in the *fault-free* case. Higher k values increase the malicious advantage. Single-provider COoL selection ($k = 1$) performs best: as long as $p_M \leq p_{exodus}^{delay}$, the IHS advantage remains within 2% of the fault-free case at 50%. p_{exodus}^{delay} is the proportion of malicious providers beyond which honest provider throughput is insufficient to serve all honest requests under *provider-side delay attacks*. When $p_M > p_{exodus}^{delay}$, i.e., with more than 5 malicious providers, honest requests are increasingly served by malicious providers, increasing in turn malicious IHS. In contrast, DeSearch-like selection with any k and COoL multiprovider selection with $k > 2$ both start with higher IHS advantage and show a steady increase with p_M.

2) Provider-side load balancing: In Figure 6b, we compare spatial Power-of-two [15] (sPoT) to our proposed solution in a *delay-attacked* TEE environment. We then also evaluate the impact of the new attack introduced by provider-side load balancing: the *queue attack* coupled with *delay attacks*.



(a) Impact of SGX and ORAM [19] on DeSearch and COoL-TEE. (b) Impact of the time source type on COoL-TEE provider selection.

Fig. 5: Mean latency versus throughput depending on enabled security and time mechanisms, with resp. $N_P = 4$ and 3.



(a) Impact of TEE *delay attacks* on single-/multi-provider selection (b) Impact of TEE-based attacks on sPoT [15] and COoL selections

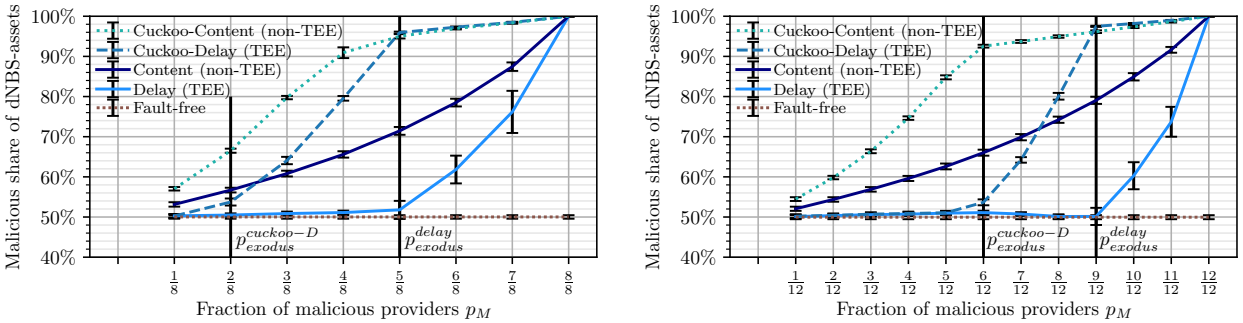
Fig. 6: Information head-start case-study with respect to competitors in a *delay-attacked* TEE environment — Shares of discovered NBSa across 100 runs, in a malicious same-datacenter environment with 8 providers. *The closer the values are to 50%, the better, indicating lower malicious IHS advantage.*

First, with only *delay attacks* and with standalone sPoT [15], the share of dNBSa by malicious consumers increases steadily with p_M. In sPoT, consumers rely on a network distance measurement (e.g., a ping) to determine their two closest providers. In an adversarial same-datacenter setup, where all providers are roughly equidistant to a given consumer, and can lie about one-shot latency measurements, sPoT amounts to single-provider random selection: "DeSearch-like-k=1" in Figure 6a and "sPoT" in Figure 6b have similar shapes. Meanwhile, with consumer-side COoL selection, as well as with its $k = 2$ variant augmented with sPoT at the provider-side, the malicious dNBSa share remains close to the fault-free case as long as $p_M \leq p^{delay}_{exodus}$.

However, when we consider the increased attack surface introduced with provider-side load balancing, i.e., when malicious providers leverage *queue attacks* on top of *delay* attacks, solutions based on this provider-side balancing suffer more from IHS. Standalone "COoL" selection is client-side only, so this attack vector does not exist.

3) COoL selection under fire: Figure 7a focuses on IHS advantage using single-provider COoL selection, when subject to different provider- and consumer-side attacks. *Content-* and *delay-based attacks* show different behaviors: while *(cuckoo-)content attacks* have the share of dNBSa grow steadily with p_M, *(cuckoo-)delay attacks* have this share close to the fault-free case, so long as p_M is less than the attack-specific threshold p_{exodus}. After their p_{exodus}, the IHS advantage rises sharply for each *delay-based attack*. Note that $p^{cuckoo-D}_{exodus} < p^{delay}_{exodus}$. Without cuckoo attacks, malicious consumers aim to minimize latency, driving them towards malicious providers, which honest consumers avoid. In *cuckoo-delay attacks*, malicious consumers prioritize loading honest providers with requests over latency minimization. Thus, honest consumers shift towards malicious providers at a lower p_M i.e., $p^{cuckoo-D}_{exodus} < p^{delay}_{exodus}$. In Figure 7a, the impact of *cuckoo-delay attacks* at a given p_M is similar to a *provider-side-only delay attack* with three additional malicious providers. Hence, a lower p_M achieves the same IHS advantage when malicious consumers actively participate in attacks.

4) Provider scale-out: Figure 7b evaluates the impact of scaling out providers in response to high loads: 4 additional providers are launched compared to Figure 7a. The incoming consumer load remains the same, reducing the total system load ρ from 75% (8 providers) to 50% (12 providers). Lowering ρ raises the minimum thresholds $p^{cuckoo-D}_{exodus}$ and p^{delay}_{exodus} for impactful attacks: $p_M > 25\%$ at $\rho = 75\%$, while $p_M > 50\%$ at $\rho = 50\%$. $p^{cuckoo-D}_{exodus}$ is tied to ρ, marking when honest consumers cannot handle the total request load: $p^{cuckoo-D}_{exodus} = 1-\rho$. p^{delay}_{exodus} depends on ρ and c_M, as only honest consumers load honest providers with requests in provider-



(a) IHS attacks on COoL provider selection with 8 total providers (b) IHS attacks on COoL provider selection, scaled out by 4 providers

Fig. 7: Information head-start case-study with respect to attack scenarios in (non-)TEE environments — Shares of discovered NBSa across 100 runs, in a malicious same-datacenter environment, with either 8 or 12 providers (resp. Figures 7a and 7b).

side-only delay attacks: $p_{exodus}^{delay} = 1 - (1 - c_M)\rho$. Without TEEs, under (cuckoo-)content attacks, malicious dNBSa shares are not sensitive to ρ: malicious IHS advantage depends on p_M for provider-side content attacks, and on p_M and c_M for cuckoo-content attacks. The slope change in Figures 7a and 7b occurs when the proportion of malicious providers equals the proportion of honest consumers: $p_{exodus}^{cuckoo-C-satur} = 1 - c_M$.

5) Trusted Time for COoL-TEE: Finally, we evaluate how measuring trustworthy delays and relaying them to consumers impacts IHS in Figure 8a. We show IHS for COoL-TEE with and without Trusted Time ("w/ TT" and "w/o TT", respectively), under provider-side delay attacks as well as cuckoo-delay attacks. "COoL w/o TT" curves are the same as in Figure 7a. Adding trusted time to COoL-TEE, resilience against IHS in provider-side delay attacks slightly improves, by up to 4% less malicious dNBSa share. Against the more powerful cuckoo-delay attack, resilience is greatly improved with trusted time: "COoL w/ TT" under cuckoo-delay attacks behaves similarly to "COoL w/o TT" under the weaker provider-side delay attacks. Similarly to previous figures, the curve's change in trend happens around p_{exodus}^{delay}.

6) Case-studies summary: To summarize and answer *RQ2*, in the same-datacenter setup, with COoL provider selection, *delay-attacked* honest consumers discover NBSa at similar rates as malicious consumers, as long as the honest providers' maximum throughput is sufficient to serve either: all requests without trusted time or all *honest* requests with trusted time. In practice, this can be ensured using load-aware elasticity mechanisms [33]: the fourth case-study in Section V-B4 and Figure 7 show how reducing the total system load ρ increases the necessary malicious provider proportion p_M for impactful attacks. Meanwhile, without TEEs, or without latency-aware provider selection, the discovery of dNBSa by malicious consumers increase nearly linearly with respect to the proportion of malicious providers p_M. Experiments were also run either with periodic asset arrivals, or periodic inter-request emission delays, but we do not illustrate them due to lack of space: conclusions drawn above also hold in those cases, albeit with higher standard deviations.

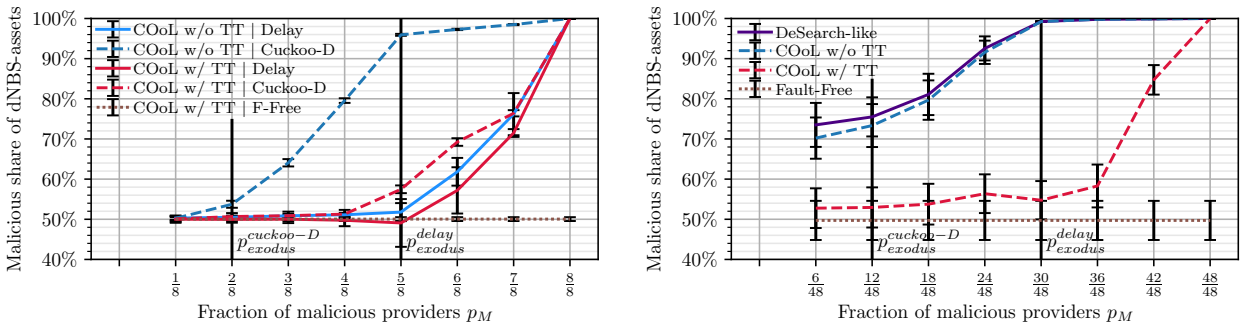
C. Adversarial multi-datacenter setup

To answer *RQ3*, we now analyze IHS in a multi-datacenter setting in Figure 8b, under a cuckoo-delay attack. The total system load is $\rho = 75\%$ like in Figure 8a. We compare both figures in the following analysis. For a fairer comparison to DeSearch's random selection in a context with geo-distributed datacenters, DeSearch consumers in this experiment only randomly select among their personal best-performing providers, i.e., among those in Algorithm 1's B_{prov} (line 6). We first observe that advantage deviations from the average are higher in the multi-datacenter setup than same-datacenter setup, due to the heterogeneous consumer-provider latencies and random sampling of consumer locations from the dataset, whereas all consumers have 0ms network latency to all providers in the same-datacenter setup. Under a cuckoo-delay attack, "DeSearch-like" and "COoL w/o TT" in Figure 8b behave similarly to "COoL w/o TT" in the same-datacenter setup (Figure 8a), but with an accentuated malicious advantage even for low malicious presence. In comparison, Figure 8b's "COoL w/ TT" remains under 7% additional malicious share for $p_M \leq p_{exodus}^{delay}$. This behavior is similar to its equivalent in the same-datacenter setup. In other words, even in the multi-datacenter setup, COoL-TEE *with* Trusted Time under the powerful two-sided attack exhibits a resilience close to that of COoL-TEE *without* Trusted Time under the weaker provider-side-only delay attack.

VI. Related work

COoL-TEE addresses issues at the intersection of multiple research domains, which we compare here.

1) Decentralized search mechanisms: Few recent works focus on enabling a more decentralized governance over search mechanisms. DHT-based solutions like HypeerCube [9] and Ditto [8] do not guarantee result completeness. TheGraph [34] indexes Ethereum smart contracts but lacks completeness guarantees. DeSearch [7] uses distributed TEEs for subsecond latencies and result integrity, but is vulnerable to IHS attacks due to the host's control over the network interfaces. While Kadabra [35] does not rely on TEEs and instead proposes a



(a) 8 providers in a single *(cuckoo-)delay-attacked* datacenter.

(b) 48 providers in multiple *cuckoo-delay-attacked* datacenters.

Fig. 8: Information head-start case-study with respect to attacks on COoL-TEE with and without Trusted Time (TT) — Shares of discovered NBSa across 100 runs, in a malicious same- or multi-datacenter environment (resp. Figure 8a and Figure 8b).

DHT-based search system based on Kademlia [36], it does not address IHS and suffers from multi-second latencies.

2) Reputation systems: COoL-TEE's provider selection policy can be seen as a reputation system, locally updated by each consumer while only cooperating with TEEs, i.e., trustlessly with respect to other protocol actors. More advanced reputation systems open many attack vectors, notably well-studied in the peer-to-peer system literature [37]–[40].

3) Decentralized QoS-based provider selection: Provider selection protocols address IHS indirectly but usually assume a non-adversarial setting. COoL-TEE adapts these mechanisms to malicious environments using TEEs. Go-with-the-winner [41] selects providers based on past round-trip latencies in a non-adversarial setting, while sPOT [15] uses proximity-based selection, which we have shown to be vulnerable to IHS. DONAR [42] introduces a mapper role to select providers, but it can be bypassed by malicious consumers, raising IHS risk.

4) Resilient time measurements with TEEs: This work focuses on leveraging Intel SGX TEEs, readily available at multiple major Cloud Service Providers. The `sgx_get_trusted_time` primitive is unavailable for Linux since the 2020 SGX SDK's version 2.9, leading previous trusted time solutions to be deprecated [43], [44] and new ones to be proposed [12], [13]. T3E [12] uses a local TPM as a source of trusted time for the TEE who continually polls it for timestamps. However, the latency for a usable timestamp in the client execution is TPM-specific, in the order of tens to hundreds of milliseconds. Moreover, T3E relies on remote users to detect a throughput drop when the host performs a delay attack between the TPM and the TEE, which is complex to implement in practice in our context, with many concurrent clients instead of a single one commissioning a task. Triad [13] uses clusters of multiple TEEs in the same datacenter to maintain a shared notion of elapsed time. New SGX processors additionally propose the AEX-Notify functionality [22], which enables developpers to implement arbitrary code to run when a TEE enclave resumes its execution after an interruption. This interrupt-awareness enables Triad to then refresh the local timestamp using either in-cluster TEEs or a remote time authority. Because time is measured inside the TEE with Triad, it offers a mean clock error in the hundreds of microseconds, within our requirements for COoL-TEE. Alder et al. [21] formalize security levels for TEE timers and survey recent solutions for the main TEE architectures, including Intel SGX and TDX, AMD SEV, as well as ARM TrustZone and CCA.

VII. Conclusion

In this paper, we highlight how providers of a marketplace's distributed search service are able to give favorable treatment to part of the userbase, through *information head-start* attacks, even if Trusted Execution Environments (TEEs) are used to guarantee the integrity and confidentiality of the search mechanism.

We show that quality-of-service-aware provider selection as performed by our proposed mechanism COoL-TEE, on top of TEEs, is paramount to counter such malicious provider behavior. Thanks to COoL-TEE, we show that whether users collude or not with malicious providers, they do not get an edge over others in terms of fresher market state information, as long as there is enough honest provider throughput to serve all honest requests.

References

[1] J. Greig, "Europol identifies hundreds of e-commerce platforms used in digital skimming attacks," 2023. [Online]. Available: https://therecord.media/europol-identifies-hundreds-ecommerce-skimmers

[2] A. Cuthbertson, "Facebook down: Users report issues with Messenger and Instagram," 2021. [Online]. Available: https://www.independent.co.uk/tech/facebook-down-messenger-instagram-not-working-b1950938.html

[3] J. Swearingen, "When Amazon Web Services Goes Down, So Does a Lot of the Web," 2018. [Online]. Available: https://nymag.com/intelligencer/2018/03/when-amazon-web-services-goes-down-so-does-a-lot-of-the-web.html

[4] "Censorship by Google," 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Censorship_by_Google&oldid=1220502028

[5] P. Garg, "Steemit Censoring Users on 'Immutable' Social Media Blockchain," 2019. [Online]. Available: https://cryptoslate.com/steemit-censoring-users-immutable-blockchain-social-media/

[6] A. Glaser, "How Apple and Amazon Are Aiding Chinese Censors," 2017. [Online]. Available: https://slate.com/technology/2017/08/apple-and-amazon-are-helping-china-censor-the-internet.html

[7] M. Li, J. Zhu, T. Zhang, C. Tan, Y. Xia, S. Angel, and H. Chen, "Bringing Decentralized Search to Decentralized Services," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021, pp. 331–347.

[8] N. V. Keizer, O. Ascigil, M. Król, and G. Pavlou, "Ditto: Towards decentralised similarity search for web3 services," in *2023 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2023, pp. 66–75.

[9] M. Zichichi, L. Serena, S. Ferretti, and G. D'Angelo, "Towards Decentralized Complex Queries over Distributed Ledgers: a Data Marketplace Use-case," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, Jul. 2021, pp. 1–6.

[10] S. Eskandari, S. Moosavi, and J. Clark, "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain," in *Financial Cryptography and Data Security*, A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala, Eds. Springer International Publishing, 2020, pp. 170–189.

[11] C. Baum, J. H.-y. Chiang, B. David, T. K. Frederiksen, and L. Gentile, "SoK: Mitigation of Front-Running in Decentralized Finance," in *Financial Cryptography and Data Security. FC 2022 International Workshops - CoDecFin, DeFi, Voting, WTSC, Grenada, May 6, 2022, Revised Selected Papers*, ser. Lecture Notes in Computer Science, S. Matsuo, L. Gudgeon, A. Klages-Mundt, D. P. Hernandez, S. Werner, T. Haines, A. Essex, A. Bracciali, and M. Sala, Eds., vol. 13412. Springer, 2022, pp. 250–271. [Online]. Available: https://doi.org/10.1007/978-3-031-32415-4_17

[12] G. M. Hamidy, P. Philippaerts, and W. Joosen, "T3E: A Practical Solution to Trusted Time in Secure Enclaves," in *Network and System Security*, ser. Lecture Notes in Computer Science, S. Li, M. Manulis, and A. Miyaji, Eds. Springer Nature Switzerland, 2023, pp. 305–326.

[13] G. Fernandez, A. Brito, and C. Fetzer, "Triad: Trusted Timestamps in Untrusted Environments," in *2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2023, pp. 169–176. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10475818

[14] M. Bettinger, "COoL-TEE source code," https://github.com/RedChainLab/COoL-TEE, 2024.

[15] N. K. Panigrahy, T. Vasantam, P. Basu, D. Towsley, A. Swami, and K. K. Leung, "On the analysis and evaluation of proximity-based load-balancing policies," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 7, no. 2-4, pp. 1–27, 2022.

[16] C. Stathakopoulou, S. Rüsch, M. Brandenburger, and M. Vukolić, "Adding fairness to order: Preventing front-running attacks in bft protocols using tees," in *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2021, pp. 34–45.

[17] M. Kelkar, S. Deb, S. Long, A. Juels, and S. Kannan, "Themis: Fast, Strong Order-Fairness in Byzantine Consensus," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '23. Association for Computing Machinery, 2023, pp. 475–489. [Online]. Available: https://dl.acm.org/doi/10.1145/3576915.3616658

[18] V. Costan and S. Devadas, "Intel SGX explained," *Cryptology ePrint Archive*, 2016. [Online]. Available: https://eprint.iacr.org/2016/086

[19] X. Wang, H. Chan, and E. Shi, "Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. Association for Computing Machinery, 2015, pp. 850–861. [Online]. Available: https://doi.org/10.1145/2810103.2813634

[20] S. Cen and B. Zhang. (2019) Trusted Time and Monotonic Counters with Intel® Software Guard Extensions Platform Services. Intel. [Online]. Available: https://www.intel.com/content/www/us/en/content-details/671564/trusted-time-and-monotonic-counters-with-intel-software-guard-extensions-platform-services.html

[21] F. Alder, G. Scopelliti, J. Van Bulck, and J. T. Mühlberg, "About Time: On the Challenges of Temporal Guarantees in Untrusted Environments," in *Proceedings of the 6th Workshop on System Software for Trusted Execution*, ser. SysTEX '23. Association for Computing Machinery, 2023, pp. 27–33. [Online]. Available: https://doi.org/10.1145/3578359.3593038

[22] S. Constable, J. V. Bulck, X. Cheng, Y. Xiao, C. Xing, I. Alexandrovich, T. Kim, F. Piessens, M. Vij, and M. Silberstein, "{AEX-Notify}: Thwarting Precise {Single-Stepping} Attacks through Interrupt Awareness for Intel {SGX} Enclaves," 2023, pp. 4051–4068. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/constable

[23] E. S. Raymond, "Ntpsec: A secure, hardened NTP implementation," *Linux Journal*, vol. 2016, no. 270, 2016.

[24] Network Time Foundation. (2022) Clock filter and discipline algorithms. NTP: Network Time Protocol. [Online]. Available: https://www.ntp.org/documentation/4.2.8-series/{filter,discipline}/

[25] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

[26] M. Fiducioso, S. Curi, B. Schumacher, M. Gwerder, and A. Krause, "Safe contextual bayesian optimization for sustainable room temperature pid control tuning," *arXiv preprint arXiv:1906.12086*, 2019.

[27] W. Xu, Y. Jiang, B. Svetozarevic, P. Heer, and C. N. Jones, "Config: Constrained efficient global optimization for closed-loop control system optimization with unmodeled constraints," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 513–518, 2023.

[28] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[29] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2010.

[30] B. Charyyev, E. Arslan, and M. H. Gunes, "Latency Comparison of Cloud Datacenters and Edge Servers," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9322406

[31] RIPE Atlas - RIPE Network Coordination Centre. [Online]. Available: https://atlas.ripe.net/

[32] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001.

[33] Kubernetes. (2024) Horizontal Pod Autoscaling. [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[34] The Graph, 2018, website available at: https://thegraph.com/.

[35] Y. Zhang and S. B. Venkatakrishnan, "Kadabra: Adapting Kademlia for the Decentralized Web," in *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part II*, ser. Lecture Notes in Computer Science, F. Baldimtsi and C. Cachin, Eds., vol. 13951. Springer, 2023, pp. 327–345. [Online]. Available: https://doi.org/10.1007/978-3-031-47751-5_19

[36] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Springer, 2002, pp. 53–65.

[37] A. S. Almasoud, F. K. Hussain, and O. K. Hussain, "Smart contracts for blockchain-based reputation systems: A systematic literature review," vol. 170, p. 102814, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S108480452030285X

[38] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-Based Distributed Trust and Reputation Management Systems: A Survey," vol. 8, pp. 21 127–21 151, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8970496/?arnumber=8970496

[39] O. Hasan, L. Brunie, and E. Bertino, "Privacy-Preserving Reputation Systems Based on Blockchain and Other Cryptographic Building Blocks: A Survey," vol. 55, no. 2, pp. 32:1–32:37, 2022. [Online]. Available: https://doi.org/10.1145/3490236

[40] R. H. Pereira, M. J. Gonçalves, and M. A. G. Magalhães, "Reputation Systems: A framework for attacks and frauds classification," vol. 8, no. 1, p. 19218, 2023. [Online]. Available: https://www.jisem-journal.com/article/reputation-systems-a-framework-for-attacks-and-frauds-classification-12830

[41] C. Liu, R. K. Sitaraman, and D. Towsley, "Go-with-the-winner: Performance based client-side server selection," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2016, pp. 404–412.

[42] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," in *Proceedings of the ACM SIGCOMM 2010 conference*, 2010, pp. 231–242.

[43] S. Tople, S. Park, M. S. Kang, and P. Saxena, "VeriCount: Verifiable Resource Accounting Using Hardware and Software Isolation," in *Applied Cryptography and Network Security*, B. Preneel and F. Vercauteren, Eds. Springer International Publishing, 2018, pp. 657–677.

[44] F. M. Anwar, L. Garcia, X. Han, and M. Srivastava, "Securing Time in Untrusted Operating Systems with TimeSeal," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019, pp. 80–92. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9052115