CACTEE: Confidential Asset Certification using Trusted Execution Environments

Istemi Ekin Akkus
istemi_ekin.akkus@nokia-bell-labs.com
Nokia Bell Labs
Stuttgart, Germany

Ivica Rimac
ivica.rimac@nokia-bell-labs.com
Nokia Bell Labs
Stuttgart, Germany

Abstract

Collaboration ecosystems present a crucial opportunity for novel applications in industrial, enterprise and consumer contexts by enabling organizations to share various datasets, models and software. This sharing, however, creates potential security problems (e.g., tampered datasets, poisoned models, software supply chain attacks). Efforts like Software/Machine Learning Bill of Materials (SBOMs, MLBOMs) and open-sourcing focus on ownership and accountability by making the assets and their dependencies more transparent. Unfortunately, the inherent conflict between transparency and confidentiality limits the extent of collaborations. Compliance to regulations further complicates these efforts.

In this paper, we advocate that Trusted Execution Environments (TEEs) can help address these issues. We present a general system, **CACTEE**, that creates third-party verifiable certificates of various properties for a variety of confidential assets. Besides the reproducible and persistent properties of the assets, **CACTEE** also supports ephemeral, unreproducible properties of the computation (e.g., location, energy usage), which can help with compliance-related issues.

CCS Concepts: • **Security and privacy → Trusted computing**; **Distributed systems security**.

Keywords: TEE, Confidential Asset, Certification, Proof of Property, Third-party Verifiable, Datasets, Models, Software

ACM Reference Format:
Istemi Ekin Akkus and Ivica Rimac. 2026. CACTEE: Confidential Asset Certification using Trusted Execution Environments. In *9th Workshop on System Software for Trusted Execution (SysTEX '26), April 27–30, 2026, Edinburgh, Scotland Uk.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3805690.3805734

1 Introduction

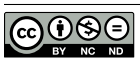
Collaboration ecosystems present a crucial opportunity for novel solutions and applications in industrial, enterprise and consumer contexts [8, 24, 45, 63, 104]. These ecosystems allow entities to share with partners various assets, enabling them to enhance their existing products or develop new ones. Examples of such assets include datasets, machine learning (ML) models and software.

This setup, however, creates potential problems regarding security. Software supply chain attacks [21] are a common occurence [60, 67, 109]. ML solutions increasingly become more prevalent in many applications and critical infrastructure [34, 77], creating additional attack surfaces, such as poisoned datasets or backdoored models [15, 38]. These security problems negatively affect the collaboration opportunities. Such issues, rooted in the lack of trust among these entities, are usually addressed through contractual agreements among partners and audits by trusted third parties. These solutions, however, are usually manual, error-prone and costly.

To address these problems and to improve automation and security, some efforts focus on accountability by creating a provenance trail of assets with increased transparency. Software Bill of Materials (SBOMs), introduced in 2021 [17, 117], specifically focuses on open-source software. Recent efforts extend the same idea to datasets and ML models with ML Bill of Materials (MLBOMs) [22, 25, 26]. For transparency, these assets declare their usage of open-source software packages with their versions, licenses and vulnerabilities as well as a description of build processes. The assets are then supplied by well-known owners with their digital signatures.

While these efforts help with security and automation, transparency requirements conflict with a crucial property: confidentiality of the assets. Such assets require time, money and resources to create: datasets need to be cleaned and curated, models need to be designed and trained, software needs to be developed and maintained. Consequently, organizations owning such assets may want to monetize them, but making them public for transparency directly conflicts with this goal. Furthermore, in heavily-regulated industries like healthcare and finance, complying with various privacy laws, like GDPR [9] and EU AI Act [33], makes such transparency efforts more complicated, if not infeasible.

In addition, recipients of an asset may be interested in its properties *before* it is available to them, to see if it satisfies their use case requirements [2]. Although many properties are *persistent* and easily reproducible, some important properties may not necessarily belong to the asset but to the *computation* that produced the asset. Such properties are



SysTEX '26, Edinburgh, Scotland Uk

ACM ISBN /2026/04
https://doi.org/10.1145/3805690.3805734

ephemeral, existing only *during* the computation. For example, certain privacy regulations (e.g., GDPR [9], EU AI Act [33]) dictate where data must be processed; thus, it is important to determine and record the location while the computation is running. Likewise, the energy usage of a computation (e.g., model training) must be captured at run time. A provenance trail of an asset (i.e., SBOM) is useful for accountability, but cannot pro-actively prevent a requirement being violated (e.g., compliance to regulations).

In this paper, we advocate that Trusted Execution Environments (TEEs) enable an ideal solution to the challenges of confidentiality and ephemerality. Because of their integrity protections, TEEs allow potential users of the assets to have more confidence in the advertised properties. We present **CACTEE** that enables automated generation of *asset certificates* for a variety of *confidential* assets, including datasets, ML models and software. Any third party (e.g., potential buyers, regulatory auditors) can verify these certificates, even after the computation has finished. The certificates not only record persistent properties of the assets but also the ephemeral properties of the computations. Using confidential GPUs (e.g., NVIDIA H100 [81]), **CACTEE** can also support large-scale assets and computations, and is available as open-source [12].

In the next section, we describe several motivational use cases, and how **CACTEE** helps alleviate their challenges. We describe persistent and ephemeral properties in §3, with two important examples of jurisdiction and energy usage. §4 presents our assumptions, threat model and goals. We present the design of **CACTEE** in §5, followed by our implementation and evaluation in §6 showcasing various certification examples, involving datasets, ML models and software. We present related work in §7, and conclude with discussion and future work in §8.

2 Motivational Use Cases

Here, we present a few motivational use cases, which revolve around protecting confidential assets while increasing trustworthiness and addressing compliance-related issues.

Marketplaces. Marketplaces for exchanging datasets, models and software [8, 24, 45, 63, 104] provide organizations with a great opportunity to obtain crucial pieces for their applications. Using such an asset in critical infrastructure [34, 77] requires trustworthiness in the asset's properties (e.g., anonymized for GDPR [9]) and its provider (e.g., reputable provider). However, providers may not always want to open-source their confidential assets for transparency [45, 63] or expose them to marketplace experts for vetting [24, 104]. Verifiable asset certificates address both problems at once: Potential users can check if a given asset satisfies desired properties with increased confidence, while providers can advertise their assets without losing confidentiality.

Collaborative ML. In many industries, some organizations hold useful training data (e.g., banks, hospitals, operators) and others critical expertise to develop models (e.g., FinTech, pharma, equipment manufacturers). The sensitivity of these assets require these organizations to take confidentiality seriously, at the expense of collaboration. While TEEs help facilitate collaborations with increasing confidentiality [16, 78, 79, 87, 120], utility and security problems arise: dataset owners need confidence that others are providing their fair share; model owners need confidence the datasets are useful and compliant to regulations [9, 33]. Verifiable asset certificates reduce time and effort in creating collaborations by allowing checks of such requirements *beforehand*.

Software Supply Chain. Although open-source constitutes a big chunk of today's software, in many cases software is still distributed in binary format without its source (e.g., office applications, tax software). In addition, much of the software used today has many dependencies, making it a complex process to deal with vulnerabilities [21, 60, 67, 109]. To alleviate some of these problems, SBOMs [17, 117] and MLBOMs [22, 25, 26] focus on the provenance and transparency of the software [50, 102, 112] by utilizing owner signatures and secure hashes of the software, so that users can verify its authenticity [17, 26, 101]. Although these efforts help with *post-mortem* analyses by determining and holding the responsible entities accountable, they do not proactively prevent potential issues. Verifiable asset certificates enhance and complement this provenance information by embedding information about the build processes of confidential software, allowing organizations to make better due-diligence decisions on which software they should use, integrate and depend on [85], even with confidential assets.

Golden Values. When a service offers its users confidentiality and privacy using TEEs, it is crucial that the users can access the source, reproduce the software and compute its cryptographic hash [28, 110]. However, this is not always possible due to the confidentiality of the backend service (e.g., Apple's Private Compute Cloud [5]). As a result, users have to trust the provider when publishing the service's *golden value* [5]. Golden values published with verifiable asset certificates, referring to a well-known build process, would give more confidence to users about the backend service's integrity and correctness without exposing its confidential source.

Infeasible Reproducibility. Like distributing software binaries or golden values, statements about an asset may not be easily reproducible, even if the asset is public: the reproduction may simply require too many computational resources or too much time [10, 46, 84]. For example, just confirming that a model was trained with particular datasets may require extensive amounts of training. Verifiable asset certificates about these computations can help increase confidence in the resulting assets (e.g., model, binary) without having to replicate resource- and time-intensive computations.

Compliance to Regulations. Many legal frameworks, laws and regulations (e.g., GDPR and AI Act in EU [9, 33], CCPA, HIPAA and others in US [13, 42, 69, 80]), define various requirements for organizations handling sensitive data. Other regulations for cybersecurity (e.g., SBOM [17], Cyber Resilience Act [30]) require certain actions when developing, maintaining and selling software (e.g., due diligence on integrated components, vulnerability scans [85]). Showing applicable regulations have been followed is critical to avoid hefty penalties and gain customer trust. Current auditing practices, whereby human experts from trusted third parties access confidential assets and understand their processing [114, 115], are time-consuming, costly and error-prone. Verifiable asset certificates enable organizations to show regulators that necessary steps were taken for compliance via technical and cryptographic means, without exposing their confidential assets to human auditors. They also enable auditing to be automated and scalable, making it fast and cheap.

3 Persistent and Ephemeral Properties

Different use cases, contexts and regulatory frameworks may have various requirements for an asset. *Persistent* properties of an asset do not change and can be reproduced after it is obtained (e.g., statistics of a purchased dataset). In contrast, *ephemeral* properties exist *during* a computation, meaning they cannot be reproduced even if the asset is later available and unchanged. In other words, the property belongs not to the asset but to the computation. These properties usually accompany the computation of a persistent property, and can be computed with the same code, regardless of use case. As such, they can help with various compliance-related topics if recorded during the computation and can be verified even after the computation is finished. Below, we describe two important examples.

Jurisdiction of Computation via Geolocation. Besides defining requirements on why data is collected, processed, stored and used, regulations like GDPR [9] and EU AI Act [33] also describe restrictions on *where* data is being processed. CCPA [13], HIPAA [42] and others [69, 80] define various conditions to be satisfied (e.g., secure storage, encryption), forcing organizations to structure their resources according to applicable laws. Being able to show regulations were followed becomes critical, requiring to determine *where* data was processed, so that its jurisdiction and thus applicable laws and regulations can be ascertained and enforced. As such, it becomes crucial to capture and record this ephemeral information during the computation in a verifiable manner.

Energy Usage. Many studies show how AI/ML is driving the demand in energy, not just for training new models using GPUs [89, 90, 107] but also for serving them with support infrastructure (e.g., storage, cooling) [83]. The ongoing debate about AI's impact on sustainability leads to initiatives and regulations for activities of corporations [31–33, 43, 116], producing a need for organizations to show their impact when handling confidential assets (e.g., models). As such, energy usage, ephemerally available only *during* a computation, needs to be captured and recorded in a verifiable manner.

4 Goals, Actors, & Assumptions

4.1 Goals

Our main goal is to enable automated generation of verifiable certificates about persistent and ephemeral properties for a variety of confidential assets. The discussed target use cases in §2 involve other parties (e.g., buyers, auditors); thus, the generated certificates should be *third-party verifiable*. To support large-scale assets (e.g., datasets, models) and long computations (e.g., model training, software compilation), the system should exploit accelarators like GPUs for *scalability*.

4.2 Actors

The **service provider** is responsible for deploying and exposing the service that creates verifiable asset certificates in automated and scalable fashion. An **asset owner** directly interacts with the service to obtain a third-party verifiable certificate for a given asset about its properties. Asset owners can be organizations monetizing their assets, participating in collaborations or creating confidential services for their users. The asset certificates enable them to gain customer trust and demonstrate compliance. An **asset user** is interested in the properties of an asset. Asset users can be potential buyers in a marketplace, organizations determining to collaborate with others, or auditors. They receive and verify the certificates to check if the properties satisfy their use case requirements or conditions for compliance to regulations.

4.3 Assumptions & Threat Model

We assume that the TEE guarantees are intact. TEE manufacturers like Intel take a vigilant approach regarding vulnerabilities and their patches [53]. Nonetheless, we leave physical, side-channel, and other attacks on TEEs outside our scope. Without such attacks, any collusion between asset owners and the service provider becomes moot, and we assume there is no such collusion. Similarly, we assume that the TEE platform software is up-to-date; cloud providers invest significantly into confidential computing, in some cases running their own cloud services using TEEs [52].

Although the assets may be confidential, the code operating on the assets, the Property Computation Code (PCC) and service code, are publicly available (e.g., open-source). Similarly, any libraries used in the PCC and service code are assumed to be available (e.g., Ubuntu OS, packages). If a package's source is not available, we assume that it is signed by its owner and trusted (e.g., NVIDIA H100 drivers).

We assume the PCC is a self-contained archive (e.g., `tar` with files owned by `root`) that can be built into a container

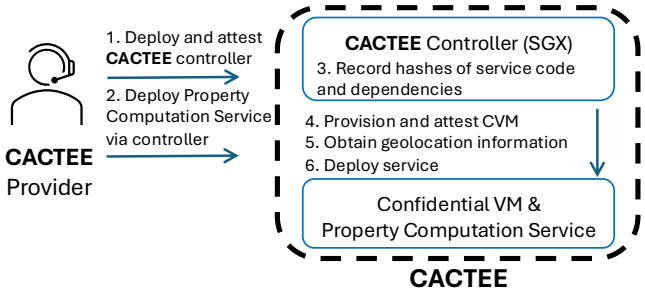


Figure 1. CACTEE deployment by provider.

image (i.e., has a `Dockerfile`). Such an archive's secure hash can be computed deterministically (unlike Docker image layers). It is also much smaller than the actual container image.

5 CACTEE Design

We first give an overview of our building blocks and their composition in **CACTEE**, followed by the workflows to deploy the service, use it to generate certificates and then verify them. We also describe how **CACTEE** captures two important ephemeral properties of geolocation and energy usage.

5.1 Building Blocks

CACTEE uses TEEs, such as Intel SGX [55], Intel TDX [57], AMD SEV-SNP [4] and NVIDIA H100 [81]. These TEEs can be on-premise with standard remote attestation procedures [51, 54, 61] or provided by cloud providers (e.g., Azure, Google, AWS) with their own attestation services [3, 40, 72].

Enclaves + CVMs. duet [1, 108] employs an application enclave (e.g., Intel SGX) as a trustworthy controller (via its quote [54]). The controller acts as an orchestrator for confidentiality-offering services deployed on CVMs, and enables their runtime customization with a log of administrative actions (e.g., installing packages, configuring services) to complement the initial CVM attestation reports not reflecting such information [61]. In duet, CVMs are only administered by the controller, preventing 'rogue owner' attacks.

Enclave-signed output. TEEs usually target two-entity scenarios, where a relying party verifies an attested TEE service. To provide third-party verifiability, we adopt the approach proposed by PraaS [2, 93], which generates a public/private keypair at the initialization of an application enclave and embeds the public key as part of its attestation quote. At the end of a computation, the corresponding private key is used to sign the output, which can be verified by any user with the attestation quote, even after the enclave is destroyed.

5.2 Overview

Although duet's approach fits our scenario, its generic logic supporting multiple services simultaneously and its use of action logs for establishing trust in CVM runtime state creates complexity and scalability challenges in certificate generation and verification workflows. By combining the enclave-signed output with a customized controller enclave, **CACTEE** generalizes asset certificate generation for a variety of confidential assets (e.g., datasets, models, software) and a variety of computations (e.g., model training, software compilation). In **CACTEE**, the controller serves a single service for confidential asset certification. Instead of keeping runtime action logs for CVMs, the **CACTEE** controller exposes these actions as part of its source code, so that the secure measurement reflects them in the attestation quote [54]. These changes make generation and verification of asset certificates easier and more scalable.

CACTEE supports capturing and recording ephemeral properties of the computations like geolocation and energy usage. Ephemeral properties indicate that a particular computation has been performed with the asset and are computed by **CACTEE** directly. In contrast, persistent properties of an asset are computed by the potentially custom Property Computation Code (PCC) used in the certification.

Note that the *accurate* computation of ephemeral properties is orthogonal to our design; their logic can be separately designed, implemented, evaluated, and integrated into **CACTEE** as modules. While developing such modules is important, it is outside this paper's scope and future work. We demonstrate **CACTEE**'s capability to capture such properties with proof-of-concept approaches.

Rationale. We could use reproducible builds [95] that embed the service logic within a tamper-proof CVM image [37, 118] or Intel TDX's Runtime Measurement Register for CVM runtime state [65, 91, 119]. These approaches require a new check and distribution of trustworthy measurement values when service code changes. In contrast, a duet-like approach allows extending the service code in the CVM with additional tools (e.g., energy usage) without changing the controller, reusing the checks about its correctness.

5.3 CACTEE Service Deployment

The provider first deploys the **CACTEE** controller as an SGX application and performs remote attestation with it [54] via an attestation service (e.g., Microsoft Azure Attestation [74, 75], Intel Attestation Service [56]) (Step 1 in Figure 1). The provider also checks for the expected measurement value (`MRENCLAVE`) in the quote. Afterwards, the provider deploys the property computation service by uploading its code and dependencies to the controller (Step 2), which records their secure hashes as part of its *service metadata* (Step 3). The service dependencies here refer to a script of various package installations and configurations performed on the CVM at runtime. The controller then provisions one (or more) CVMs with a well-known VM image (e.g., Ubuntu 22.04) and performs remote attestation to ensure their correctness [61, 74] (Step 4) as well as captures and records their geolocation (Step 5). Finally, the controller installs the dependencies on

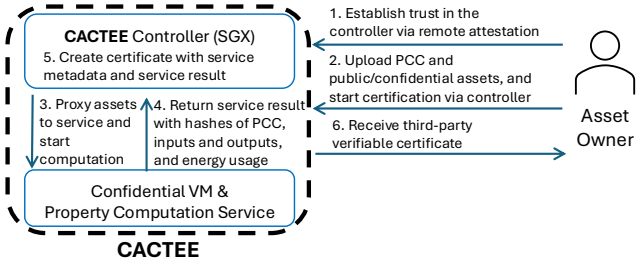


Figure 2. Asset owner workflow for certification.

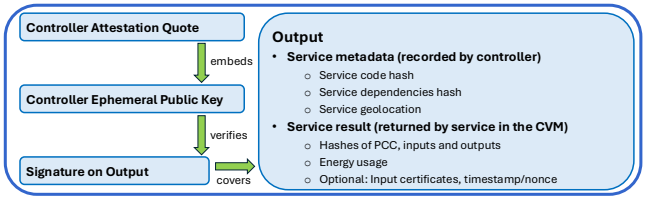


Figure 3. Third-party verifiable asset certificate.

the CVMs, sets up the TLS certificate for the service and starts it (Step 6). This controller-created certificate enables secure communication between the controller and the service in the CVM. Note that the controller is the only entity, itself protected by a TEE, that has administrative access to the CVM [1], which can be verified because the controller's code is open-source, including CVM provisioning, access control setup and certificate creation. In **CACTEE**, the provider can only start/stop CVMs and deploy the service, which is more restrictive than the original duet controller [1].

5.4 Asset Certificate Generation

After **CACTEE** is deployed, asset owners use its API to trigger certification for their confidential assets. An asset owner first performs remote attestation [54] with the controller (Step 1 in Figure 2). The controller also returns the deployed service's metadata, including its location (§5.7.1) and secure hashes of the service code and its dependencies, which are compared with the expected values (e.g., open-source, reproducible). After establishing trust, the asset owner uploads the confidential (or public) assets and public PCC over a secure connection, and triggers the certification (Step 2).

The controller proxies all uploaded inputs (i.e., assets, PCC) to a chosen CVM, where a service instance is running and starts computation (Step 3). The service computes the secure hashes of the inputs and PCC archive, and builds the container image using the PCC. Any dependencies, reflected in the PCC's inspectable `Dockerfile`, are downloaded during the build over secure connections.

The service then runs the container image without network access with two mapped volumes: one with read-only access to inputs (`/tmp/inputs`), and one with read-write access for outputs (`/tmp/outputs`). Any property to be certified is written by the PCC to the outputs volume. After PCC's finish, the service gets the output files, computes their secure hashes and returns the *service result* (i.e., secure hashes, energy usage, input certificates) to the controller (Step 4), which creates the verifiable asset certificate (Step 5).

5.5 Certificate Content

A certificate has enough information to establish the *chain of trust* for the asset properties (Figure 3). The chain starts with the controller's SGX quote. The quote includes the controller's reproducible secure measurement value (`MRENCLAVE`) and the secure hash of its ephemeral public key generated at its start (`REPORTDATA`) [2]. The corresponding private key is used to produce a signature inside the controller enclave.

This signature covers both the *service metadata* and *service result*. The service metadata contains the geolocation information (§5.7.1) and the secure hashes of the service code and its dependencies. The service result includes the secure hashes of PCC inputs, PCC outputs and the PCC as well as energy usage (§5.7.2). It also optionally contains any certificates of the inputs (e.g., dataset certificates for model training) and a timestamp/nonce (e.g., for on-demand certification scenarios).

5.6 Asset Certificate Verification

The asset certificate can be verified by any user (e.g., potential buyer, auditor). After obtaining the certificate from the asset owner (e.g., via marketplaces, website), the asset user first checks the TEE quote's validity via an attestation service (e.g., Microsoft Azure Attestation [74, 75], Intel Attestation Service [56]) and the secure measurement value (`MRENCLAVE`). Then, the controller public key is extracted and used to verify its signature on the output. Afterwards, the asset user compares the service metadata (i.e., location, hashes of the service code and dependencies) and PCC output hashes with their expected values. If verified, the asset user has confidence that the properties in the PCC outputs were computed in a TEE. Finally, the user checks if those properties satisfy use case or regulatory requirements.

5.7 Capturing Ephemeral Properties

Here, we present how **CACTEE** captures two important ephemeral properties with our proof-of-concept approaches.

5.7.1 Geolocation for Jurisdiction. Physically locating a machine on the internet is an old problem. Most related work relies on latency measurements to well-known landmarks and multilateration [29, 39, 48, 59, 121]. IP geolocation databases like MaxMind [70] provide geolocation even at street-level, but may not work with virtual machines due to internal networks of cloud providers. For our proof-of-concept, we employ multiple techniques together, matching results indicating a higher confidence in the geolocation. Our insight is that country-level geolocation estimations are

Algorithm 1 Geolocation via latency measurements.

```
1: procedure MeasureGeolocation()
2:     measurements ← ∅
3:     anchors ← collectAnchorsInfoCities()
4:     loop for each city ∈ anchors
5:         repAnchor ← getRandomAnchor(area)
6:         RTTs ← measureRTTs(repAnchor)
7:         measurements[repAnchor] ← min(RTTs)
8:     closestAnchors ← getClosest(measurements, n)
9:     return closestAnchors
```

enough for jurisdiction. In addition, different use cases may require different trust levels: in some scenarios, the cloud provider's declaration of location may be acceptable.

As such, **CACTEE** records: **(i)** the CVM region set in **CACTEE** deployment, **(ii)** the cloud provider metadata of the provisioned CVM, **(iii)** multiple IP geolocation database lookups for the CVM's IP address, and **(iv)** the latency measurements from inside the CVM to RIPE ATLAS anchors [97]. These latency measurements can then be utilized to estimate the geolocation of the service.

Algorithm 1 shows our high-level geolocation procedure. We first collect information about RIPE ATLAS anchors [96] and group them into cities (Line 3). For each city, we pick a random anchor as the *representative*, measure multiple round-trip times (RTT) to it and store the minimum RTT (Lines 4-7), in which we observe that delays may change due to congestion, but physical distances do not. In other words, we optimistically assume there is no congestion during our measurements. Afterwards, we pick n (e.g., 20) smallest RTT values of all representative anchors (Line 8) and return them with their locations to be recorded by the controller (Line 9). For an estimate, the anchor locations can be used to compute a weighted average of (`lat`, `long`) coordinates or probabilities for countries (e.g., more weights for smaller RTTs).

Potential Limitations: If no anchors are in the same country or the CVM is located in a border area, the country estimation may fail. A malicious cloud provider can redirect CVM traffic through proxies, inevitably increasing the RTTs. These RTTs can be compared with reference values from the (faked) location to known locations. Faked RTT responses can be prevented with TLS-based RTT requests [111]. As future work, we plan to compare our measurement values with reference measurements between two well-known locations. which can also be performed at the verifying party.

5.7.2 Energy Usage. With increasing awareness of the energy demand of ML workloads [58, 64, 83, 107], recent projects attempt to measure and quantify their energy usage and carbon emissions [14, 19, 35, 36, 92]. Some efforts usually require access to hardware sensors like Intel RAPL [35, 92], which may not be possible in the cloud. As a proof-of-concept in **CACTEE**, we utilize CodeCarbon [19] because it works in cloud environments by estimating the power usage of CPUs and GPUs with available vendor tools (e.g., NVIDIA, Intel, Apple) and known hardware specifications (e.g., Thermal Design Power) [20]. For each certification related API call, we initialize CodeCarbon and accumulate its output. These calls include uploading assets and code, downloading URL-referred assets, building the PCC container image and running it.

Potential Limitations: The lack of direct hardware sensor access in a VM causes our information to be an estimate. Note that estimating the energy usage outside **CACTEE** would require additional trust assumptions on the host, where the CVM and the property computation service is running.

6 Implementation & Evaluation

6.1 Implementation

We developed **CACTEE** controller based on duet [108] and modified it for our goals (e.g., single service, geolocation). Our modified **CACTEE** controller consists of about 2.5K lines of documented Python code, and is containerized using Gramine libOS [41, 113]. We implemented the property computation service, energy usage estimation and a utility tool for easy access to HuggingFace repos [45] in about 1K lines of documented Python code. We also developed a client SDK to interact with **CACTEE** and verify certificates, again in about 1K lines of documented Python code. In addition, we developed a local testing tool that utilizes the same application logic as the property computation service to ease development and debugging of Property Computation Code (PCC). Using it, we developed several PCC examples, some of which are generic and can be applied to multiple assets (e.g., extract statistical properties), whereas others are customized (e.g., specific benchmarks for an ML model). Our implementation and certification examples are available as open-source [12].

6.2 Deployment Setup

We deployed **CACTEE** on Azure Cloud with an Intel SGX capable machine for the controller (i.e., `DCsv3` with 2 vCPUs and 16GB RAM) and an AMD SEV-SNP machine with NVIDIA H100 (i.e., `NCC40ads_H100_v5` with 40 vCPUs and 320GB RAM). We utilize Microsoft Azure Attestation for remote attestation for both the controller and the CVM (via the controller) along Azure's guidelines [75].

6.3 Certification Examples

Without loss of generality, we use open-source assets for ease of availability; we do not assume an asset is public. Because PCCs are customizable, **CACTEE** can create verifiable certificates for a variety of assets, including different types of datasets (e.g., image, audio, text, code), ML models and software, with the same underlying service (Table 1). The certificate size depends on the number of input and output files of the PCC. It contains the secure hashes of each file, so

Table 1. Asset certification examples. All sizes include only inputs (i.e., no dependencies). ML model sizes include also datasets. Certification time and energy usage are approximate.

Asset Type	Asset Name	PCC Description	Asset Size (MB)	Certificate Size (KB)	Certification Time (h:m:s)	Certification Energy Usage (Wh)
Dataset (image)	ILSVRC/imagenet-1k [49] validation set	Compute image metadata	6451	14	00:08:30	9.5
Dataset (image)	COCO [18] 2017 validation set	Count images with various objects using YOLOv3 [62, 94]	778	14	00:07:24	8.3
Dataset (audio)	AquaV/fallout-4-voices [6]	Extract audio metadata	268	130	00:40:59	15.7
Dataset (text)	Open-Orca/OpenOrca [84]	Count unique system prompts	3164	14	00:02:40	2.6
Dataset (text)	open-thoughts/OpenThoughts-114k [86]	Extract column summaries	3635	17	00:02:41	1.7
Dataset (code)	bigcode/the-stack-dedup [10]	Count licenses and comment languages in Rust & Swift code	6207	19	02:58:38	197.8
ML Model	HuggingFaceTB/SmolLM-135M [46]	Compute GPQA [47] benchmark	2168	25	01:05:19	119.0
ML Model	sentence-transformers/all-MiniLM-L6-v2 [99]	Compute several benchmarks	1224	178K	00:27:25	35.9
ML Model	sgugger/glue-mrpc [100]	Compute loss, accuracy, F1 score	434	24	00:11:00	12.4
Software	`RT_PREEMPT kernel` for Rasp. Pi [98]	Build RT kernel deb packages	236	14	00:59:01	67.6
Software	Bootstrap 5.3.8 [11]	Minimize & package Bootstrap	8.1	20	00:02:32	2.6
Software	NVIDIA CUDA 13.1.1 container image [82] nvidia/cuda:13.1.1-cudnn-devel-ubuntu24.04	Generate SBOM, check vulnerabilities and licenses	3788	15	00:07:40	8.5

a recipient can check if the acquired assets match the certificate that was shown. In addition, **CACTEE** takes advantage of the H100 GPU [81] for different types of computations (e.g., benchmarking, compilation) and for large-scale assets (e.g., datasets, ML models). Even with open-source assets, reproducing the claimed properties would require significant amount of resources, time and energy, highlighting the benefits of verifiable asset certificates.

7 Related Work

Transparency. With increasing interest in SBOMs and ML-BOMs [25, 26], there are more tools [22, 23, 76, 101, 105] and end-to-end security approaches available. in-toto [50, 112] proposes a framework, whereby one can specify how and by whom a software artifact can be built and integrated from its source to its final form. Atlas [106] proposes a lifecycle management system for ML datasets and models, employing TEEs to record various training parameters and operations. While these efforts provide auditable trails for artifacts, they require asset transparency (e.g., code, datasets, training parameters). **CACTEE** complements these efforts by producing third-party verifiable certificates for confidential assets.

Hardware-rooted proofs. PraaS [2] allows generation of verifiable proofs of properties for datasets. Laminator [27] provides a system for verifiable model cards for some properties. Both systems use Intel SGX [55]; thus, have scalability issues due to memory limits and lack of GPU support, limiting their use to small datasets and models. SLSA [102] gives adoptable guidelines for increasing supply chain security via levels with increasing protections. Hardware-Attested Build Environments [44, 71, 103] use secure hardware modules for ensuring that a software artifact's build process is not tampered with. **CACTEE** generalizes this integrity protection to also include confidential datasets and models.

Trustworthy collaborations. TEES have been also used in creating confidential collaboration environments, like data clean rooms [7, 68, 73]. **CACTEE**'s verifiable asset certificates can help facilitate such collaborations and give participants confidence about their confidential assets of their partners. Verifiable asset certificates can be also utilized as part of the outputs to demonstrate that they were generated inside such an environment.

8 Discussion & Future Work

We advocated that TEEs can help solve problems arising from the conflict between transparency to improve trustworthiness and confidentiality of assets, like datasets, ML models and software. We presented **CACTEE** that provides third-party verifiable certificates about persistent and ephemeral properties that apply to multiple use cases, complement transparency efforts and address compliance concerns, without sacrificing the asset confidentiality. We also showed its effectiveness with large-scale assets and computations. Our code and examples are available as open-source [12].

We created a verifiable certificate for SBOM generation to prevent tampering by the asset owner [66, 88]. As future work, we plan to embed verifiable SBOM generation in CI/CD pipelines. Asset certificates of listed packages can also be put in SBOMs with SPDX [105] or CycloneDX [23] extensions.

Another line of future work is developing better ways of capturing ephemeral geolocation and energy usage information. For better geolocation information, compiling and maintaining trustworthy latency measurement values as reference measurements between known locations may address some of the limitations of our proof-of-concept approach.

While defining a formal mapping between legal requirements of regulations and TEE security guarantees of verifiable asset certificates is outside this paper's scope, we envision that the technical and cryptographic protections TEEs offer can be utilized in the same line as expert witnesses. Understanding and formalizing such a mapping is also an avenue of future work.

Acknowledgements

We thank the reviewers for their insightful comments to improve this paper. We also thank the members of our research group for helping develop PCCs for our examples.

References

[1] Istemi Ekin Akkus and Ivica Rimac. 2024. Duet: Combining a Trustworthy Controller with a Confidential Computing Environment. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*.

[2] Istemi Ekin Akkus, Ivica Rimac, and Ruichuan Chen. 2024. PraaS: Verifiable Proofs of Property as-a-Service with Intel SGX. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*.

[3] Amazon. [n. d.]. *Attestation with AMD SEV-SNP - Amazon Elastic Compute Cloud*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/snp-attestation.html

[4] AMD. 2020. *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More*. https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[5] Apple Security Engineering and Architecture (SEAR), User Privacy, Core Operating Systems (Core OS), Services Engineering (ASE), and Machine Learning and AI (AIML). 2024. *Private Cloud Compute: A new frontier for AI privacy in the cloud*. https://security.apple.com/blog/private-cloud-compute/

[6] AquaV. [n. d.]. *AquaV/fallout-4-voices - Datasets at Hugging Face*. https://huggingface.co/datasets/AquaV/fallout-4-voices

[7] AWS. [n. d.]. *Data Collaboration Service - AWS Clean Rooms - AWS*. https://aws.amazon.com/clean-rooms/

[8] AWS. [n. d.]. *Data Marketplace - AWS Data Exchange - AWS*. https://aws.amazon.com/data-exchange/

[9] Ben Wolford. [n. d.]. *What is GDPR, the EU's new data protection law? - GDPR.eu*. https://gdpr.eu/what-is-gdpr/

[10] bigcode. [n. d.]. *bigcode/the-stack-dedup - Datasets at Hugging Face*. https://huggingface.co/datasets/bigcode/the-stack-dedup

[11] Bootstrap. [n. d.]. *Bootstrap - The most popular HTML, CSS and JS library in the world*. https://getbootstrap.com/

[12] CACTEE authors. 2026. *Nokia-Bell-Labs/confidential-asset-certification-using-tees*. https://github.com/Nokia-Bell-Labs/confidential-asset-certification-using-tees

[13] California Office of the Attorney General. [n. d.]. *California Consumer Privacy Act | State of California - Department of Justice - Office of the Attorney General*. https://www.oag.ca.gov/privacy/ccpa

[14] CarbonTracker. [n. d.]. *saintslab/carbontracker*. https://github.com/saintslab/carbontracker

[15] Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning Web-Scale Training Datasets is Practical. In *2024 IEEE Symposium on Security and Privacy (SP)*. 407–425. doi:10.1109/SP54263.2024.00179

[16] Dong Chen, Alice Dethise, Istemi Ekin Akkus, Ivica Rimac, Klaus Satzke, Antti Koskela, Marco Canini, Wei Wang, and Ruichuan Chen. 2025. Protecting Confidentiality, Privacy and Integrity in Collaborative Learning. arXiv:2412.08534 [cs.DC] https://arxiv.org/abs/2412.08534

[17] CISA. [n. d.]. *Software Bill of Materials (SBOM) | CISA*. https://www.cisa.gov/sbom

[18] COCO. [n. d.]. *COCO datasets: 2017 validation set*. https://cocodataset.org/#download

[19] Codearbon Authors. [n. d.]. *mlco2/codecarbon*. https://github.com/mlco2/codecarbon

[20] CodeCarbon authors. [n. d.]. *Methodology - CodeCarbon 3.2.1 documentation*. https://mlco2.github.io/codecarbon/methodology.html

[21] Cybersecurity and Infrastructure Security Agency. 2021. *Defending Against Software Supply Chain Attacks*. https://www.cisa.gov/sites/default/files/publications/defending_against_software_supply_chain_attacks_508_1.pdf

[22] CycloneDX. -. *Machine Learning Bill of Materials (ML-BOM) | CycloneDX*. https://cyclonedx.org/capabilities/mlbom/

[23] CycloneDX. [n. d.]. *CycloneDX Bill of Materials Standard | CycloneDX*. https://cyclonedx.org

[24] Databricks. [n. d.]. *Databricks Marketplace | Databricks*. https://www.databricks.com/product/marketplace

[25] David Mor-Ofek. 2024. *It's Time to Talk About AI/ML BOM (Artificial Intelligence Bill of Materials) And Vulnerability Management*. https://c2a-sec.com/its-time-to-talk-about-ai-ml-bom-artificial-intelligence-bill-of-materials-and-vulnerability-management/

[26] Diana Kelley. 2024. *Why MLBOMs Are Useful for Securing the AI/ML Supply Chain*. https://www.darkreading.com/vulnerabilities-threats/mlboms-are-useful-for-securing-ai-ml-supply-chain

[27] Vasisht Duddu, Lachlan J. Gunn, and N. Asokan. 2025. Laminator: Verifiable ML Property Cards using Hardware-assisted Attestations. In *Proceedings of the Fifteenth ACM Conference on Data and Application Security and Privacy* (Pittsburgh, PA, USA) *(CODASPY '25)*. Association for Computing Machinery, New York, NY, USA, 317–328. doi:10.1145/3714393.3726492

[28] Edgeless Systems. [n. d.]. *Privatemode AI - The always encrypted AI service*. https://www.privatemode.ai/

[29] Mojtaba Eskandari, Anderson Santana De Oliveira, and Bruno Crispo. 2014. VLOC: An Approach to Verify the Physical Location of a Virtual Machine In Cloud. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. 86–94. doi:10.1109/CloudCom.2014.47

[30] European Commission. 2024. *Cyber Resilience Act | Shaping Europe's digital future*. https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act

[31] European Commission. 2025. *Corporate sustainability reporting*. https://finance.ec.europa.eu/capital-markets-union-and-financial-markets/company-reporting-and-auditing/company-reporting/corporate-sustainability-reporting_en

[32] European Commission. [n. d.]. *Corporate sustainability due diligence*. https://commission.europa.eu/business-economy-euro/doing-business-eu/sustainability-due-diligence-responsible-business/corporate-sustainability-due-diligence_en

[33] European Parliament and of the Council. 2024. *Regulation (EU) 2024/1689*. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

[34] Ewelina Gregolinska, Rehana Khanam, Frédéric Lefort, Prashanth Parthasarathy. 2022. *Capturing the true value of Industry 4.0*. https://www.mckinsey.com/capabilities/operations/our-insights/capturing-the-true-value-of-industry-four-point-zero

[35] Guillaume Fieni, Daniel Romero Acero, Pierre Rust, and Romain Rouvoy. 2024. PowerAPI: A Python framework for building software-defined power meters. *Journal of Open Source Software* 9, 98 (June 2024), 6670. doi:10.21105/joss.06670

[36] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. 2020. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. In *CCGRID 2020 - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*. Melbourne, Australia. doi:10.1109/CCGrid49817.2020.00-45

[37] Anna Galanou, Khushboo Bindlish, Luca Preibsch, Yvonne-Anne Pignolet, Christof Fetzer, and Rüdiger Kapitza. 2023. Trustworthy confidential virtual machines for the masses. In *Proceedings of the 24th International Middleware Conference (Middleware '23)*. Association for Computing Machinery.

[38] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Mądry, Bo Li, and Tom

Goldstein. 2023. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 2 (2023), 1563–1580. doi:10.1109/TPAMI.2022.3162397

[39] Mark Gondree and Zachary N.J. Peterson. 2013. Geolocation of data in the cloud. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy* (San Antonio, Texas, USA) *(CODASPY '13)*. Association for Computing Machinery, New York, NY, USA, 25–36. doi:10.1145/2435349.2435353

[40] Google. [n. d.]. *Confidential VM attestation | Google Cloud.* https://cloud.google.com/confidential-computing/confidential-vm/docs/attestation

[41] Gramine. [n. d.]. *GitHub - gramineproject/gramine: A library OS for Linux multi-process applications, with Intel SGX support.* https://github.com/gramineproject/gramine

[42] HHS. [n. d.]. *Summary of the HIPAA Privacy Rule | HHS.gov.* https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html

[43] Tim Hickman, Thomas Burmeister, Erasmus Hoffmann, Petra Karin Iffert, and Aishwarya Jha. 2025. *Energy efficiency requirements under the EU AI Act.* https://www.whitecase.com/insight-alert/energy-efficiency-requirements-under-eu-ai-act

[44] Daniel Hugenroth, Mario Lins, René Mayrhofer, and Alastair R Beresford. 2025. Attestable builds: compiling verifiable binaries on untrusted systems using trusted execution environments. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 4514–4528.

[45] Hugging Face. [n. d.]. *Hugging Face - The AI community building the future.* https://huggingface.co/

[46] HuggingFaceTB. [n. d.]. *HuggingFaceTB/SmolLM-135M - Hugging Face.* https://huggingface.co/HuggingFaceTB/SmolLM-135M

[47] Idavidrein. [n. d.]. *Idavidrein/gpqa - Datasets at Hugging Face.* https://huggingface.co/datasets/Idavidrein/gpqa

[48] Said Ider and Maryline Laurent. 2025. GeoFINDR: Practical Approach to Verify Cloud Instances Geolocation in Multicloud. arXiv:2504.18685 [cs.NI] https://arxiv.org/abs/2504.18685

[49] ILSRVC. [n. d.]. *ILSRVC/imagenet-1k validation set.* https://image-net.org/download.php

[50] in-toto. [n. d.]. *in-toto: A framework to secure the integrity of supply chains.* https://in-toto.io/

[51] Intel. [n. d.]. *Attestation Service for Intel Software Guard Extensions (Intel SGX): API Documentation.* https://www.intel.com/content/dam/develop/public/us/en/documents/sgx-attestation-api-spec.pdf

[52] Intel. 2024. *Microsoft Deploys Confidential Computing To Protect $25B per Year in Customer Payments.* https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2024-02/microsoft-azure-confidential-computing-solution-brief.pdf

[53] Intel. [n. d.]. *Affected Processors: Transient Execution Attacks & Related Security...* https://www.intel.com/content/www/us/en/developer/topic-technology/software-security-guidance/processors-affected-consolidated-product-cpu-model.html

[54] Intel. [n. d.]. *Attestation Services for Intel Software Guard Extensions.* https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html

[55] Intel. [n. d.]. *Intel Software Guard Extensions (Intel SGX).* https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/software-guard-extensions.html

[56] Intel. [n. d.]. *Intel® Trust Authority.* https://www.intel.com/content/www/us/en/security/trust-authority.html

[57] Intel. [n. d.]. *Intel® Trust Domain Extensions (Intel TDX).* https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html

[58] International Energy Agency. 2025. *Energy and AI.* https://www.iea.org/reports/energy-and-ai

[59] Chetan Jaiswal and Vijay Kumar. 2016. IGOD: identification of geolocation of cloud datacenters. *Journal of Information Security and Applications* 27-28 (2016), 85–102. doi:10.1016/j.jisa.2016.01.001 Special Issues on Security and Privacy in Cloud Computing.

[60] Jen Easterly, Tom Fanning. 2023. *The Attack on Colonial Pipeline: What We've Learned & What We've Done Over the Past Two Years.* https://www.cisa.gov/news-events/news/attack-colonial-pipeline-what-weve-learned-what-weve-done-over-past-two-years

[61] Jeremy Powell. 2022. *AMD SEV-SNP Attestation: Establishing Trust in Guests.* https://www.amd.com/content/dam/amd/en/documents/developer/lss-snp-attestation.pdf

[62] Joseph Redmon. [n. d.]. *YOLO: Real-Time Object Detection.* https://pjreddie.com/darknet/yolo/

[63] Kaggle. [n. d.]. *Kaggle: Your Machine Learning and Data Science Community.* https://www.kaggle.com/

[64] Mahmut Kandemir. 2025. *Why AI uses so much energy — and what we can do about it.* https://iee.psu.edu/news/blog/why-ai-uses-so-much-energy-and-what-we-can-do-about-it

[65] Kevin Wang. 2025. *TEE Attestation Guide for dstack Applications.* https://github.com/Dstack-TEE/dstack/blob/master/attestation.md#tee-attestation-guide-for-dstack-applications

[66] JungA Kim, Yiseul Choi, and Seongmin Kim. 2025. Attestation-Based SBOM Integrity Verification for Secure and Transparent Software Supply Chains. In *2025 25th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 1–4. doi:10.23919/APNOMS67058.2025.11181389

[67] Lucian Constantin. 2020. *SolarWinds attack explained: And why it was so hard to detect.* https://www.csoonline.com/article/570191/solarwinds-supply-chain-attack-explained-why-organizations-were-not-prepared.html

[68] ManaTEE Project authors. [n. d.]. *ManaTEE Project.* https://github.com/manatee-project/manatee

[69] Matt Schwartz. 2025. *Consumer Reports and EPIC Laud Massachusetts Senate for Unanimously Passing Strong Privacy Bill.* https://advocacy.consumerreports.org/press_release/consumer-reports-and-epic-laud-massachusetts-senate-for-unanimously-passing-strong-privacy-bill/

[70] MaxMind. [n. d.]. *Industry leading IP Geolocation and Online Fraud Prevention | MaxMind.* https://www.maxmind.com/en/home

[71] Marcela S. Melara and Chad Kimes. 2024. Auditing the CI/CD Platform: Reproducible Builds vs. Hardware-Attested Build Environments, Which is Right for You?. In *Proceedings of the 2024 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '24)*. Association for Computing Machinery, New York, NY, USA, 43–44. doi:10.1145/3689944.3696351

[72] Microsoft. [n. d.]. *About Azure confidential VMs | Microsoft Learn.* https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview

[73] Microsoft. [n. d.]. *Azure Confidential Clean Rooms Preview.* https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-clean-rooms

[74] Microsoft. [n. d.]. *GitHub - Azure/confidential-computing-cvm-guest-attestation: Confidential VM Platform Guest attestation sample apps.* https://github.com/Azure/confidential-computing-cvm-guest-attestation

[75] Microsoft. [n. d.]. *What is guest attestation for confidential VMs? | Microsoft Learn.* https://learn.microsoft.com/en-us/azure/confidential-computing/guest-attestation-confidential-vms

[76] mithril-security. [n. d.]. *AICert.* https://aicert.mithrilsecurity.io/en/latest/

[77] Mithun Nagabhairava, Daniel Pender, Joe Wagner. 2023. *How AI, ML and Industry 4.0 affect plant automation.* https://www.plantengineering.com/articles/how-ai-ml-and-industry-4-0-affect-plant-automation/

[78] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services*. 94–108.

[79] Fan Mo, Zahra Tarkhani, and Hamed Haddadi. 2024. Machine Learning with Confidential Computing: A Systematization of Knowledge. *ACM Comput. Surv.* 56, 11, Article 281 (June 2024), 40 pages. doi:10.1145/3670007

[80] New York State Attorney General. 2019. *The SHIELD Act.* https://ag.ny.gov/resources/organizations/data-breach-reporting/shield-act

[81] NVIDIA. [n. d.]. *Confidential Compute on NVIDIA Hopper H100.* https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/HCC-Whitepaper-v1.0.pdf

[82] NVIDIA. 2026. *nvidia/cuda - Docker Image.* https://hub.docker.com/r/nvidia/cuda

[83] James O'Donnell and Casey Crownhart. 2025. *We did the math on AI's energy footprint. Here's the story you haven't heard.* https://www.technologyreview.com/2025/05/20/1116327/ai-energy-usage-climate-footprint-big-tech/

[84] Open-Orca. [n. d.]. *Open-Orca/OpenOrca - Datasets at Hugging Face.* https://huggingface.co/datasets/Open-Orca/OpenOrca

[85] Open Regulatory Compliance Working Group. [n. d.]. *https://cra.orcwg.org/faq/official/manufacturers/due-diligence/.* https://cra.orcwg.org/faq/official/manufacturers/due-diligence/

[86] open-thoughts. [n. d.]. *open-thoughts/OpenThoughts-114k - Datasets at Hugging Face.* https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k

[87] Wojciech Ozga, Do Le Quoc, and Christof Fetzer. 2021. Perun: Confidential Multi-stakeholder Machine Learning Framework with Hardware Acceleration Support. In *Data and Applications Security and Privacy XXXV*, Ken Barker and Kambiz Ghazinour (Eds.). Springer International Publishing, Cham, 189–208.

[88] Can Ozkan, Xinhai Zou, and Dave Singelee. 2025. Supply Chain Insecurity: The Lack of Integrity Protection in SBOM Solutions. arXiv:2412.05138 [cs.CR] https://arxiv.org/abs/2412.05138

[89] Pratyush Patel, Esha Choukse, Chaojie Zhang, Íñigo Goiri, Brijesh Warrier, Nithish Mahalingam, and Ricardo Bianchini. 2024. Characterizing Power Management Opportunities for LLMs in the Cloud. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 207–222. doi:10.1145/3620666.3651329

[90] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs.LG] https://arxiv.org/abs/2104.10350

[91] Phala. 2025. *Understanding TDX Attestation Reports: A Developer's Guide.* https://phala.com/posts/understanding-tdx-attestation-reports-a-developers-guide

[92] PowerAPI. [n. d.]. *powerapi-ng/powerapi.* https://github.com/powerapi-ng/powerapi

[93] PraaS authors. 2024. *Nokia-Bell-Labs/proof-as-a-service.* https://github.com/Nokia-Bell-Labs/proof-as-a-service

[94] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. arXiv:1804.02767 [cs.CV] https://arxiv.org/abs/1804.02767

[95] Reproducible Builds Contributors. [n. d.]. *Reproducible Builds - a set of software development practices that create an independently-verifiable path from source to binary code.* https://reproducible-builds.org/

[96] RIPE. [n. d.]. *REST API Reference | RIPE ATLAS Documentation.* https://atlas.ripe.net/docs/apis/rest-api-reference/

[97] RIPE. [n. d.]. *RIPE Atlas - Dashboard.* https://atlas.ripe.net/

[98] ros-realtime. [n. d.]. *Build RT_PREEMPT kernel for Raspberry Pi.* https://github.com/ros-realtime/linux-real-time-kernel-builder

[99] sentence-transformers. [n. d.]. *sentence-transformers/all-MiniLM-L6-v2 - Hugging Face.* https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[100] sgugger. [n. d.]. *sgugger/glue-mrpc - Hugging Face.* https://huggingface.co/sgugger/glue-mrpc

[101] Sigstore. [n. d.]. *Trust & Security - Sigstore.* https://www.sigstore.dev/

[102] SLSA. [n. d.]. *SLSA - Supply-chain Levels for Software Artifacts.* https://slsa.dev/

[103] SLSA collaborators. 2023. *Workstream: Hardware Attested Build Environments.* https://github.com/slsa-framework/slsa/issues/975

[104] Snowflake. [n. d.]. *Snowflake Data Marketplace | Snowflake Data Cloud.* https://www.snowflake.com/en/data-cloud/marketplace/

[105] SPDX. [n. d.]. *The System Package Data Exchange (SPDX).* https://spdx.dev

[106] Marcin Spoczynski, Marcela S. Melara, and Sebastian Szyller. 2025. Atlas: A Framework for ML Lifecycle Provenance & Transparency. In *2025 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*.

[107] Chris Stokel-Walker. 2023. *The Generative AI Race Has a Dirty Secret.* https://www.wired.com/story/the-generative-ai-search-race-has-a-dirty-secret/

[108] tee-duet authors. 2024. *Nokia-Bell-Labs/tee-duet.* https://github.com/Nokia-Bell-Labs/tee-duet

[109] Tero Mononen. 2024. *Crisis averted: A recap of the OpenSSH and XZ/liblzma incident.* https://www.ssh.com/blog/a-recap-of-the-openssh-and-xz-liblzma-incident

[110] Tinfoil. [n. d.]. *Tinfoil - Private AI.* https://tinfoil.sh/

[111] tlsping. [n. d.]. *tlsping — a tool for measuring TLS handshake latency.* https://github.com/airnandez/tlsping

[112] Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. 2019. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1393–1410. https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[113] Chia-Che Tsai, Donald E Porter, and Mona Vij. 2017. {Graphene-SGX}: A Practical Library {OS} for Unmodified Applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. 645–658.

[114] TÜV Austria. [n. d.]. *TÜV Austria.* https://en.tuv.at/

[115] TÜV SÜD. [n. d.]. *TÜV SÜD Global Home - Add Value, Inspire Trust.* https://www.tuvsud.com/en

[116] United Nations. 2025. *Artificial intelligence: How much energy does AI use?* https://unric.org/en/artificial-intelligence-how-much-energy-does-ai-use/

[117] White House. 2021. *Improving the Nation's Cybersecurity.* https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

[118] Luca Wilke and Gianluca Scopelliti. 2024. SNPGuard: Remote Attestation of SEV-SNP VMs Using Open Source Tools. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 193–198. doi:10.1109/EuroSPW61312.2024.00026

[119] Haidong Xia, Ken Lu, Ruoyu Ying, Xiaocheng Dong, and Yanhui Zhao. [n. d.]. *Runtime Integrity Measurement and Attestation in a Trust Domain.* https://www.intel.com/content/www/us/en/developer/articles/community/runtime-integrity-measure-and-attest-trust-domain.html

[120] Chengliang Zhang, Junzhe Xia, Baichen Yang, Huancheng Puyang, Wei Wang, Ruichuan Chen, Istemi Ekin Akkus, Paarijaat Aditya, and Feng Yan. 2021. Citadel: Protecting data privacy and model confidentiality for collaborative learning. In *Proceedings of the ACM Symposium on Cloud Computing*. 546–561.

[121] Yang Zhang, Dongzheng Jia, Shijie Jia, Limin Liu, and Jingqiang Lin. 2020. Splitter: An Efficient Scheme to Determine the Geolocation of Cloud Data Publicly. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. 1–11. doi:10.1109/ICCCN49398.2020.9209651

A Open Science

The code for **CACTEE** controller, property computation service and clients as well as any scripts for service deployment, certificate generation and certificate verification workflows and automation tools are available as open-source [12]. The certification examples, including all Property Computation Code (PCC), the PCC testing environment, configuration files and inputs as well as the generated asset certificates that can be verified with our verification client are also available [12].

1. **CACTEE** controller: customized duet controller to handle a single service with restricted API [1, 108]. The controller's quote and signature on the output of the Property Computation Service constitutes the third-party verifiable certificate.
2. Property Computation Service: general service for computing properties about confidential assets. The third-party verifiable asset certificates include the output of the Property Computation Service.
3. Property Computation Service tools: utility tools for convenience (e.g., easy referring to HuggingFace [45] datasets and models).
4. Asset Certification clients: clients used for deployment, certificate generation and certificate verification.
5. Various build scripts: for containerization of the controller using Gramine [41].
6. Various scripts: automation scripts using Asset Certification Clients for one-step actions for deployment, certification and verification.
7. Property Computation Code (PCC) testing environment: locally used testing environment for PCC development.
8. Asset Certification Examples: all PCC, configuration files and inputs as well as generated asset certificates (that can be verified with our verification client) used in this paper.