Attestable Audits: Verifiable AI Safety Benchmarks Using Trusted Execution Environments

Christoph Schnabl [1] Daniel Hugenroth [1] Bill Marino [1] Alastair R. Beresford [1]

Abstract

Benchmarks are important measures to evaluate safety and compliance of AI models at scale. However, they typically do not offer verifiable results and lack confidentiality for model IP and benchmark datasets. We propose Attestable Audits, which run inside Trusted Execution Environments and enable users to verify interaction with a compliant AI model. Our work protects sensitive data even when model provider and auditor do not trust each other. This addresses verification challenges raised in recent AI governance frameworks. We build a prototype demonstrating feasibility on typical audit benchmarks against Llama-3.1.

1. Introduction

Audits are an essential tool in the modern AI safety landscape as models become more capable (Aschenbrenner, 2024) and potentially more dangerous (Barrett et al., 2023; Anthropic, 2023; OpenAI, 2024), particularly in agentic environments (Chan et al., 2023). Recognizing these risks, several AI regulations (Parliament & Union, 2024; Office, 2024; The White House, 2023), policy initiatives, and AI principles (House of Commons, 2024; Solaiman, 2023; Kapoor et al., 2024) have mandated audits, but decision-makers often lack the technical information needed to evaluate auditing tools (Reuel et al., 2025). This creates a critical gap between policy and implementation that Technical AI Governance aims to close through tools, such as verifiable audits (Reuel et al., 2025). However, current audits rely on contracts or manual processes, and verification remains challenging due to restricted model access and data privacy concerns (Longpre et al., 2024a;b; Carlini et al., 2024; Cen & Alur, 2024). Furthermore, misaligned incentives between stakeholders can result in audits that do not serve the public interest (Casper et al., 2024; Raji et al., 2020; Mökander, 2023), including data exfiltration by involved actors (Eriksson et al., 2025) or models that intentionally underperform during evaluations (van der Weij et al., 2025).

To address these challenges, we investigate how users can verify they are interacting with a compliant AI system under realistic constraints: when model providers do not share weights, auditors only share code and data with regulators, and systems run on untrusted third-party infrastructure.

We propose Attestable Audits (§3), a three-step verification protocol, where auditors and model providers securely load models, audit code, and datasets into hardware-backed Trusted Execution Environments (TEEs), run benchmarks, and cryptographically attest and publish results to a public registry for user verification. We use TEEs from Confidential Computing (CC, §2) to isolate execution and encrypt memory. We demonstrate through a prototype (§5) based on AWS Nitro Enclaves that benchmarks yield expected results at 2.2× the cost of CPU and 21.7× that of GPU inference.

2. Confidential Computing (CC)

Confidential Computing (CC) ensures that critical systems protect data-in-transit, data-at-rest, and data-in-use. This is achieved by TEEs, privileged execution modes supported by modern CPUs—conceptually, a small, shielded, encrypted computer inside a computer. Backed by secure hardware, TEEs prevent interference by the host/hypervisor and encrypt all memory to thwart even physical attacks by attackers. This makes CC attractive for deployments at otherwise untrusted cloud service providers (CSPs) as long as the vendor of the secure hardware is trusted (Chen et al., 2023).

In contrast to first generation process-based TEEs (e.g., Intel SGX, Arm TrustZone), second generation TEEs (e.g., AMD SEV-SNP, Intel TDX, AWS Nitro) support full VMs (Costan, 2016; Pinto & Santos, 2019; Intel, 2025; AMD, 2025). This helps them overcome resource limitations, especially memory, that previously made ML workloads difficult (Mo et al., 2024). Whereas many CC applications focus on confidentiality properties, our work also leverages its integrity guarantees which can provide zero-knowledge-proof-like guarantees (Russinovich et al., 2024).

[1]Department of Computer Science and Technology, University of Cambridge. Correspondence to: Christoph Schnabl <cs2280@cst.cam.ac.uk>.

Workshop on Technical AI Governance (TAIG) at ICML 2025, Vancouver, Canada. Copyright 2025 by the author(s).

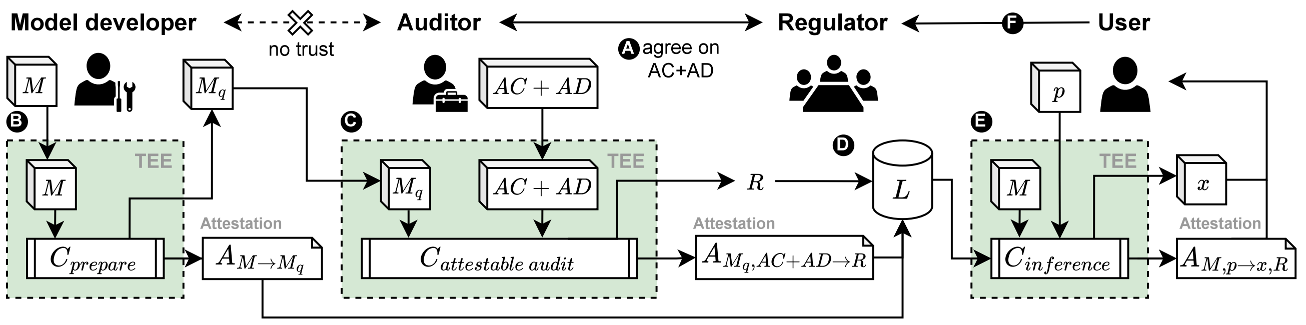

Figure 1. Overview of the Attestable Audit protocol. **A** The auditor and regulator agree on audit code AC and dataset AD. **B** Optionally, the provider prepares a (quantized) version M_q of model M, verifiable via attestation $A_{M \to M_q}$. **C** In the audit, the auditor loads encrypted $AC + AD$ into a fresh TEE and the provider loads encrypted M. **D** The audit result R and attestation $A_{Mq,AC+AD \to R}$ are published to a transparency log L. **E** The user confidentially sends prompt p to M (or M_q) in a TEE, receiving x and an attestation verifying provenance and M's compliance. **F** The user may disclose $(p, x, A_{M,p \to x,R})$ to the regulator to show audit deficits.

Clients can verify they are talking to a service inside a TEE through Remote Attestation (RA). In RA, the secure chip signs a chain of measurements, called Platform Configuration Registers (PCRs), using a non-extractable secret key. The PCRs cover the Trusted Computing Base (TCB), consisting of firmware and the loaded enclave base image. Including the enclave base image enables revocation when vulnerabilities, e.g., side-channels (Li et al., 2021), active attacks (Schlüter et al., 2024), and memory aliasing (De Meulemeester et al., 2025), are discovered.

We use AWS Nitro Enclaves (AWS, 2024) as the CC platform for our prototype. Our protocol is compatible with other CC platforms, such as Intel TDX and AMD SEV-SNP, but we leave those implementations for future work. We expect that alternatives allow for smaller TCBs and lower overhead. Importantly, these can also integrate with GPUs featuring CC support, such as Nvidia's H100 (Apsey et al., 2023) to allow for larger models. However, these eventually face limits, e.g., there is no multi-GPU support. As such, our conservative choice of a smallest common-denominator technology, ensures our design supports these challenges.

3. Attestable Audits

We present Attestable Audits as a three-step design depicted in Figure 1. First, model providers may prepare their model. Then, providers and auditors load the audit code and data into a TEE, which runs benchmarks and publishes attested results. Finally, users receive output attestations to verify interaction with an audited model.

3.1. Security Goals and Threat Model

The efficacy of the proposed system relies on the following security goals. We require **model verifiability (G1)**, i.e., the attestation includes hashes of model weights and code, and **audit verifiability (G2)**, i.e., outputs are bound to an approved audit version. The system must maintain **confidentiality (G3)** of model weights (protecting IP) and audit data to prevent "cheating". These guarantees are hard to achieve in non-CC setups but enable more robust audits, especially for closed-source systems. **Transparency (G4)** requires publishing base image, model, and audit digests, with verifiable build steps. Finally, the system must enforce **statelessness (G5)** to prevent prompt residue and covert channels (Shumailov et al., 2025), and **output verifiability (G6)** to authenticate model responses during interaction.

We assume the existence of **Network adversaries (A1)** who can intercept, tamper with, or spoof communication between components, but exclude DoS attacks. For **Physical and Privileged adversaries (A2)**, with capabilities such as RAM snapshots, VM rollbacks, and side-channel attacks.

3.2. Requirements

We use three standard cryptographic primitives available in libraries like LIBSODIUM. First, we require a pre-image and collision-resistant hashing function. Second, we require an IND-CCA secure key encapsulation mechanism (KEM) to allow two parties to share a symmetric key. The receiver generates $(pk, sk) \leftarrow$ KEM.KEYGEN() and shares pk. Then sender uses pk to generate a key and ciphertext $c, k \leftarrow$ KEM.ENCAPSULATE(pk). The receiver recovers k using KEM.DECAPSULATE(sk, c). Thirdly, we require an IND-CCA secure encryption scheme (AEAD) with $c_x \leftarrow$ AEAD.ENCRYPT(k, x) to encrypt plaintext x under key k, and $x \leftarrow$ AEAD.DECRYPT(k, c_x) to decrypt.

Furthermore, we rely on functionality provided by the TEE. First, we require attestation $(\{d, \dots\}, \text{PCR}, \sigma) \leftarrow$

Table 1. Overview of common AI safety benchmarks. Underlined benchmarks were chosen as representative $AC+AD$ for our evaluation.

Type	**Benchmarks**
Discrete-Label	MMLU (Hendrycks et al., 2020), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019)
Text-Similarity	XSum (Narayan et al., 2018), NarrativeQA (Kociský et al., 2017), CNN/DailyMail (Hermann et al., 2015)
Classifier-Judged	ToxicChat (Lin et al., 2023), BBQ (Parrish et al., 2022), RealToxicityPrompts (Gehman et al., 2020)
Retrieval	MSMARCO (Bajaj et al., 2018), Natural Questions (Kwiatkowski et al., 2019)
LLM-as-a-Judge	Chatbot Arena (Chiang et al., 2024)

$\text{ATTEST}(\{d, \dots\})$ against the currently running TEE image. The attestation includes (1) the platform configuration registers PCR that describe the loaded image, (2) auxiliary user-provided data $\{d, \dots\}$, and (3) a signature σ over all these signed with the TEE vendor's secret key. We denote attestations as $A_{in \to out}$, for binding hashed input $in = \text{HASH}(input)$ to hashed output $out = \text{HASH}(output)$.

We run the model code in a sandbox for isolation. Public model code can be included in the attested open-source base image, and only the weights need to remain confidential.

4. Protocols

This section describes the main protocols of our Attestable Audit scheme. Appendix A.1 describes used primitives and contains the pseudocode listings. The shown protocols omit details, e.g. replay prevention and key rotation mechanisms, that are important for real-world implementations.

The PREPARE protocol (Algorithm 1) offers model developers the ability to quantize their models in a confidential and verifiable manner. Practically, it enables the use of substantially smaller models with comparable performance. We discuss this ablation in Appendix A.4.2. Conceptually, it illustrates the general, abstract attestation-and-encryption workflow, based on KEM and AEAD, that we likewise employ in our subsequent protocols.

First the TEE boots from its secure image and generates a fresh KEM keypair pk, sk. It then attests to its boot image and the public key with a fresh attestation A to allow third parties to verify the given pk was indeed generated inside a TEE that booted a trusted image. The third-party first compares the PCR measurement against known trusted images and then verifies the signature σ using the TEE's vendor public key (or respective attestation service).

Once the authenticity of the TEE has been confirmed, the model provider calls $(k, c) \leftarrow \text{KEM.ENCAPSULATE}(pk)$ and uses k to encrypt their model M. The encrypted model c_M and the encrypted key c are then sent to the TEE which can derive the same key using sk and decrypt the model. The TEE then computes the quantized model M_q and measures boths by computing their hashes h_M and h_{M_q}.

Finally, the TEE encrypts the quantized model M_q using the same symmetric key k and sends it to the model provider. It also publishes the attestation $A_{M \to M_q}$ to a transparency log. This acts as evidence for third parties that they can accept models with the hash h_{M_q} as quantized versions of models with the hash h_M. The PREPARE step is convenient for practical deployments as some TEEs, e.g. the one for the audit, can only run against smaller models M_q. It allows the model provider to deploy the full model M while convincing others that the audit of M_q is a valid approximation.

The ATTESTABLEAUDIT protocol (Algorithm 2) runs audit code AC and audit dataset AD against a (quantized) model M_q in a confidential and verifiable manner. For this the TEE, as in the previous protocol, publishes an attestation with a KEM public key. Now both the model developer and the audit provider use it to upload the encrypted model c_{M_q} and audit code/dataset c_{AC+AD} to the TEE.

Once the TEE has received both, it will create a sandbox and executes AC against M_q using the audit dataset AD. The sandbox ensures that malicious code that is part of AC or M_q cannot interfere with the integrity of the overall TEE logic or invalidate previous measurements. After the execution, AC will output a single aggregated result R. The TEE then publishes an attestation $A_{M_q, AC, AD \to R}$ that binds the hash of the model and audit to this result R. This is then published to a transparency log.

The INFERENCE protocol (Algorithm 3) allows users to confidentially interact with model M (or M_q) confidentially while being able to receive guarantees that it has received score R against the audit $AC + AD$. Different to the previous protocols, the TEE first downloads the relevant attestation documents from the previous steps from the transparency logs and includes these in its initial attestation. That provides a baseline guarantee to the user that their later prompts are not given to a different model.

Next, the model provider loads their model M into the TEE using the familiar KEM+AEAD construction. The TEE will verify that the hash of the model matches with the value from the attestations and abort if that is not the case.

Then the user sends the encrypted prompt c_p to the TEE. The TEE then starts a sandbox with the model M and runs it against the input p yielding output x. A fresh attestation $A_{M,p \to x,R}$ then provides the user also with a evidence that links together model, audit result, prompt, and response. Optionally the user can later use this attestation to proof short-comings in the auditing process to a regulator.

5. Evaluation

We sample three AI-safety benchmarks (Table 1) to assess Attestable Audits' feasibility. With additional engineering, we could integrate larger benchmark suites, such as COMPL-AI (Guldimann et al., 2024) and HELM (Liang et al., 2023). We already employ zero-shot prompts from both. Our implementation, written in Rust through bindings to `llama.cpp` (Gerganov, 2023), runs on AWS Nitro Enclaves using Llama-3.1-8B-Instruct quantized to 4-bit to reduce main memory footprint at an accuracy penalty.

We log input/output token counts and prompt/response decoding latencies, ignoring non-decoding overhead (e.g., copying models into the enclave takes $\leq$2 minutes). We issue 500 prompts per benchmark on: (I) a `m5.2xlarge` instance running our protocol with enclaves enabled on 4 cores, (II) a version on `m5.xlarge` running on 4 cores, (III) a cost-constant version running `m5.2xlarge` on 8 cores, and (IV) a SOTA baseline hosting the same model, but in `fp16` precision on a cloud-based NVIDIA L40S, which uses roughly 90% of the available 48 GB VRAM through vLLM at a price of 0.89 USD/hour.

Feasibility We demonstrate that models in Attestable Audits achieve adequate performance (Table 2). In column (I), the quantized model's zero-shot MMLU accuracy is 51.4% (57.4% excluding unparsable responses), similar to 54.6% on a non-quantized model (IV) at 58.9% and LLaMa's 66.7% for 5-shot prompting (Touvron et al., 2023). The difference from (I–III) to (IV) is context size and precision; from (IV) to LLaMa's 66.7% is prompting. Summarization yields a mean BERT score of ≈0.47 vs. ≈0.58 for the non-quantized version. On ToxicChat, 1.78% are jailbreak attempts. The quantized model fails to refuse and produces toxic outputs in 2.4% of all test cases in (I) and 2.6% in the non-quantized case. Smaller differences between (I–III) stem from stochastic `top_p` sampling. A 4-bit quantized model slightly degrades performance in benchmarks. We provide a more detailed feasibility analysis in Appendix A.4.

Trade-Offs Running inference on CPUs incurs a cost overhead of 21.7× over GPU inference and suffers a 100× slowdown. The use of enclaves costs 2× due to having to use a larger instance compared to (II) or sacrificing cores relative to (III). Memory capacity constraints forces the use of smaller or quantized models. (III) hosts a 4-bit model for comparability, but could host a 8-bit model for higher

Table 2. Trade-offs and benchmark scores across: (I) enclave, (II) compute-constant, (III) cost-constant, vs. (IV) L40S GPU

Metric	(I)	(II)	(III)	(IV)
Price/hr ($)	0.38	0.19	0.38	0.89
Price/100k token ($)	5.80	2.61	3.01	0.12
Token/s	1.84	2.04	3.54	202.00
BERT Score	0.47	0.50	0.49	0.58
Toxicity rate (%)	2.40	2.00	1.70	2.60
Accuracy (%)	51.40	52.60	48.60	58.90

performance. Doubling the number of CPUs from (I) to (III) increases the throughput by almost 2 token/s. We expect larger instances to reduce the overhead by 2–5×.

Security Model and audit verifiability (**G1, G2**) are achieved through the CC-powered *audit step* (Algorithm 2). The RA process binds the hashes of the quantised model M_q, audit code AC, and dataset AD with the platform PCRs measurements. Confidentiality (**G3**) is end-to-end through the use of ephemeral keys inside the TEE. The AEAD channel bound to the initial attestation protects data-in-transit from a network adversary (**A1**). VM-level enclave isolation and full-memory encryption deny physical attackers (**A2**) access to data-in-use. Transparency (**G4**) follows from publishing the enclave base image, build scripts, content hashes, and the attestations $A_{M \to M_q}$, and $A_{M_q,\, AC+AD \to R}$ to L. Similar to Apple's PCC (Apple, 2025), anyone can rebuild and inspect the exact evaluation environment. For statelessness (**G5**) each user session runs in a fresh VM-enclave that starts with zeroized RAM and not persistent storage to eliminate prompt residue. Output integrity (**G6**) is guaranteed in the *interaction step* (Algorithm 3) analogously. For each prompt p the enclave returns $(x,\, A_{M,\, p \to x,\, R})$, to bind $\textsc{Hash}(M) \,\|\, p \,\|\, x \,\|\, R$. Verifying $A_{M,\, p \to x,\, R}$ against L confirms the reply is from the audited model with score R. Prompt-based model exfiltration during the user interaction step remains a residual gap (Carlini et al., 2024).

Engineering Challenges Our approach works well for smaller (in terms of tokens) hand-crafted datasets, as CPU inference limits token throughput. Memory constraints of CC technology complicates hosting larger models. Enclaves do not have persistent memory but instead rely on a memory-mapped file system. As a result, the runtime memory requirements can be a multiple of the underlying base image. Due to this expansion, large files such as model weights or datasets have to be transferred into the enclave during runtime. Cryptographic operations, e.g., when establishing encrypted channels, add complex logic as well as additional CPU demands. Many CC platforms come with limited documentation and lack easy-to-use software libraries. Additionally, a constant portion of main memory needs to be reserved for the host system and thus remains unavailable to the enclave, which is at least 64 MiB (AWS, 2024), but for (I) is closer to 500 MiB, or 1.5% of available memory.

6. Related Work

Hardware-Attested Integrity DeepAttest (Chen et al., 2019) binds models and code to TEEs for CNNs but lacks audit traceability and targets on-premise. Nevo et al. (2024) protect weights, while we extend this to audit datasets and code. OpenMined (2023) uses TEEs for evaluation but lacks generality and reproducibility. Our system integrates attestation into a transparent, regulator-facing pipeline.

Cryptographic Private Inference zkML (South et al., 2024) generates SNARKs for inference verification, but is orders of magnitude slower. FHE-LoRA (Frery et al., 2025) uses encrypted low-rank adaptation. SONNI (Sperling & Kulkarni, 2025) and Proof-of-Training (Sun & Zhang, 2023) focus on weight and data lineage, while we verify runtime behavior. PPFL (Mo et al., 2021) protects training gradients with TEEs, while we apply them to secure model evaluation.

Audit Frameworks & Governance Audit Cards (Staufer et al., 2025) show audit gaps, especially in reproducibility and verification. Mökander et al. (2023) propose a layered taxonomy without cryptographic guarantees. Grollier et al. (2024) show audits can enable fairwashing. Dong et al. (2024) and Leslie et al. (2023) focus on post-hoc safety, whereas we provide pre-deployment guarantees. Brundage et al. (2020) stress the importance of verifiable claims.

7. Discussion & Limitations

Our prototype has a high overhead in terms of runtime and costs. Large parts of this can be attributed to the CPU-based inference that is required by the underlying CC technology. A production-ready implementation using CC-compatible GPUs likely has an overhead as small as $5\times$. We leave this, and the other suggested extensions below, for future work.

Our architecture requires the participating parties to trust the hardware vendor of the CC technology, in our case AWS. However, all steps can be run on multiple CC technologies independently such that parties can later choose which attestation they trust. We note that while this increases integrity guarantees (trust any), the model and audit confidentiality is reduced, as a single broken TEE can leak the sensitive data.

Our prototype requires a quantized model due to technical limitations of the chosen CC technology. An implementation using a CC-compatible GPU can run the native Llama-3 model securely. However, a preparation step will still be necessary for larger models that do not fit on a single GPU and can accommodate other post-training steps, e.g., fine-tuning.

While we presented our work in the context of LLMs, it generalizes to other AI/ML systems. For instance, the operator of a self-driving car can use Attestable Audits to prove later, e.g., in court after an accident, that the very model that drove the vehicle at a given time has been correctly audited. Note, that our infrastructure also prevents human-mistakes such as mixing up audit results or accidentally deploying a wrong model version. It also naturally provides reproducibility for benchmarks during the scientific publication process.

8. Conclusion

Our Attestable Audits design uses TEEs to load audit code (AC), audit data (AD), and model weights (M) into an enclave, execute AI-safety benchmarks, and publish cryptographic proofs binding results to exact $AC + AD + M$ hashes. Our AWS Nitro Enclaves prototype runs three standard benchmarks at $21.7\times$ the cost of GPU inference, with a CPU-constant variant at $2\times$ overhead. Our protocol can be used for Verifiable Audits (Reuel et al., 2025) (§5.4.1) without exposing sensitive IP. Our main limitation stems from CPU inference, but we expect this overhead to reduce as GPU-capable enclaves become more readily available. By shifting AI governance from ex post enforcement to ex ante certifiable deployment, Attestable Audits reduces compliance and transaction costs of governing AI systems.

Policy Brief

This work directly addresses a topic that is critical to the enforcement of any AI regulation, or, more broadly, any AI governance policy: when regulators or auditors lack direct access to a model or dataset due to privacy or competition concerns, how can they ensure it complies with the relevant requirements? By introducing a method for verifiable benchmarking of AI systems running on third-party infrastructure, we help bridge this gap and enable AI developers to provide clear assurances about their models and datasets to regulators, auditors, or any other stakeholders — without exposing private assets such as model weights or proprietary data. By removing these barriers, our approach could incentivize more AI providers to enter regulated markets (Reuters, 2023) or, differently, to enroll in voluntary pre-deployment testing by third parties (Field, 2024). Separately, because our method can help shift the cost of benchmark-based audits from resource-constrained (Aitken et al., 2022) regulators to the AI developers who may be best-positioned to bear the expense of TEEs and, thus, the audits themselves, we lower the overhead associated with creating and enforcing AI regulations or AI governance policies. Given these benefits, future AI legislation might consider explicitly endorsing or encouraging such techniques.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback and suggestions, which helped improve this paper. Christoph is supported by Corpus Christi College, the German Academic Exchange Service, and the Studienstiftung des Deutschen Volkes. Daniel is supported by Nokia Bell Labs and Light Squares.

Impact Statement

Attestable Audits advance technical AI governance by allowing auditors, and users to securely verify AI model compliance. It mitigates risks of misuse or harmful outputs from AI systems through the use of trusted hardware that allows for rigorous audits without compromising sensitive data or proprietary models. While this enhances transparency and accountability, there remains a dependency on trusted hardware providers. Overall, we believe that Attestable Audits can reduce the risks associated with deploying powerful AI systems and contributes positively towards safer and more trustworthy machine learning applications.

References

Aitken, M., Leslie, D., Ostmann, F., Pratt, J., Margetts, H., and Dorobantu, C. Common Regulatory Capacity for AI. Technical report, The Alan Turing Institute, 2022. URL https://doi.org/10.5281/zenodo.6838946.

AMD. AMD Secure Encrypted Virtualization (SEV), 2025. https://www.amd.com/en/developer/sev.html. Last accessed April 2025.

Anthropic. Anthropic AI Risk Report, 2023. URL https://www-cdn.anthropic.com/1adf000c8f675958c2ee23805d91aaade1cd4613/responsible-scaling-policy.pdf.

Apple. Private Cloud Compute: A new frontier for AI privacy in the cloud, 2025. https://security.apple.com/blog/private-cloud-compute/. Last accessed April 2025.

Apsey, E., Rogers, P., O'Connor, M., and Nertney, R. Confidential computing on NVIDIA h100 GPUs for secure and trustworthy AI, 2023. https://developer.nvidia.com/blog/confidential-computing-on-h100-gpus\-for-secure-and-trustworthy-ai/. Last accessed April 2025.

Aschenbrenner, L. Situational Awareness: The Decade Ahead, 2024. Available at https://situational-awareness.ai/.

AWS. AWS Nitro Enclaves, 2024. https://aws.amazon.com/ec2/nitro/nitro-enclaves/. Last accessed December 2024.

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., and Wang, T. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset, 2018. URL https://arxiv.org/abs/1611.09268.

Barrett, A. M., Hendrycks, D., Newman, J., and Nonnecke, B. Actionable Guidance for High-Consequence AI Risk Management: Towards Standards Addressing AI Catastrophic Risks, 2023. URL https://arxiv.org/abs/2206.08966.

Brundage, M. et al. Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims, 2020. URL https://arxiv.org/abs/2004.07213.

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., Yona, I., Wallace, E., Rolnick, D., and Tramèr, F. Stealing Part of a Production Language Model, 2024. URL https://arxiv.org/abs/2403.06634.

Casper, S., Ezell, C., Siegmann, C., Kolt, N., Curtis, T. L., Bucknall, B., Haupt, A., Wei, K., Scheurer, J., Hobbhahn, M., Sharkey, L., Krishna, S., Von Hagen, M., Alberti, S., Chan, A., Sun, Q., Gerovitch, M., Bau, D., Tegmark, M., Krueger, D., and Hadfield-Menell, D. Black-Box Access is Insufficient for Rigorous AI Audits. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pp. 2254–2272. ACM, June 2024. doi: 10.1145/3630106.3659037. URL http://dx.doi.org/10.1145/3630106.3659037.

Cen, S. H. and Alur, R. From Transparency to Accountability and Back: A Discussion of Access and Evidence in AI Auditing, 2024. URL https://arxiv.org/abs/2410.04772.

Chan, A., Salganik, R., Markelius, A., Pang, C., Rajkumar, N., Krasheninnikov, D., Langosco, L., He, Z., Duan, Y., Carroll, M., Lin, M., Mayhew, A., Collins, K., Molamohammadi, M., Burden, J., Zhao, W., Rismani, S., Voudouris, K., Bhatt, U., Weller, A., Krueger, D., and Maharaj, T. Harms from Increasingly Agentic Algorithmic Systems. In *2023 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '23, pp. 651–666. ACM, June 2023. doi: 10.1145/3593013.3594033. URL http://dx.doi.org/10.1145/3593013.3594033.

Chen, H., Fu, C., Rouhani, B. D., Zhao, J., and Koushanfar, F. DeepAttest: an end-to-end attestation framework for deep neural networks. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, pp. 487–498, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366694. doi: 10.1145/3307650.3322251. URL https://doi.org/10.1145/3307650.3322251.

Chen, H., Chen, H. H., Sun, M., Li, K., Chen, Z., and Wang, X. A verified confidential computing as a service framework for privacy preservation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 4733–4750, 2023.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference, 2024. URL https://arxiv.org/abs/2403.04132.

Clark, C., Lee, K., Chang, M., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *CoRR*, abs/1905.10044, 2019. URL http://arxiv.org/abs/1905.10044.

Costan, V. Intel SGX explained. *IACR Cryptol, EPrint Arch*, 2016.

De Meulemeester, J., Wilke, L., Oswald, D., Eisenbarth, T., Verbauwhede, I., and Van Bulck, J. BadRAM: Practical memory aliasing attacks on trusted execution environments. In *46th IEEE Symposium on Security and Privacy (S&P)*, May 2025.

Dong, Y., Jiang, X., Liu, H., Jin, Z., Gu, B., Yang, M., and Li, G. Generalization or Memorization: Data Contamination and Trustworthy Evaluation for Large Language Models, 2024. URL https://arxiv.org/abs/2402.15938.

Eriksson, M., Purificato, E., Noroozian, A., Vinagre, J., Chaslot, G., Gomez, E., and Fernandez-Llorca, D. Can We Trust AI Benchmarks? An Interdisciplinary Review of Current Issues in AI Evaluation, 2025. URL https://arxiv.org/abs/2502.06559.

Field, H. OpenAI and Anthropic agree to let U.S. AI Safety Institute test and evaluate new models. *CNBC*, August 2024. URL https://www.cnbc.com/2024/08/29/openai-and-anthropic-agree-to-let-us\-ai-safety-institute-test-models.html. Published 3:01 PM EDT, Updated 6:01 PM EDT.

Frery, J., Bredehoft, R., Klemsa, J., Meyre, A., and Stoian, A. Private LoRA Fine-Tuning of Open-Source LLMs with Homomorphic Encryption. https://arxiv.org/abs/2505.07329, 2025. arXiv:2505.07329.

Gehman, S., Gururangan, S., Sap, M., Choi, Y., and Smith, N. A. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. *CoRR*, abs/2009.11462, 2020. URL https://arxiv.org/abs/2009.11462.

Gerganov, G. llama.cpp: Port of LLaMA model in pure C/C++, 2023. URL https://github.com/ggerganov/llama.cpp.

Grollier, X., Kazilsky, Y., et al. Cheating Automatic LLM Benchmarks: Null Models Achieve High Scores via Fairwashing. https://arxiv.org/abs/2410.07137, 2024. arXiv:2410.07137.

Guldimann, P., Spiridonov, A., Staab, R., Jovanović, N., Vero, M., Vechev, V., Gueorguieva, A.-M., Balunović, M., Konstantinov, N., Bielik, P., Tsankov, P., and Vechev, M. COMPL-AI Framework: A Technical Interpretation and LLM Benchmarking Suite for the EU Artificial Intelligence Act. *arXiv preprint arXiv:2410.07959*, 2024. URL https://arxiv.org/abs/2410.07959.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring Massive Multitask Language Understanding. *CoRR*, abs/2009.03300, 2020. URL https://arxiv.org/abs/2009.03300.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching Machines to Read and Comprehend. *CoRR*, abs/1506.03340, 2015. URL http://arxiv.org/abs/1506.03340.

House of Commons. Governance of Artificial Intelligence (AI). Technical report, House of Commons Science, Innovation and Technology Committee, 2024. URL https://committees.parliament.uk/publications/45145/documents/223578/default/.

Intel. Intel Trust Domain Extensions (Intel TDX), 2025. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html. Last accessed April 2025.

Kapoor, S., Bommasani, R., Klyman, K., Longpre, S., Ramaswami, A., Cihon, P., Hopkins, A., Bankston, K., Biderman, S., Bogen, M., Chowdhury, R., Engler, A., Henderson, P., Jernite, Y., Lazar, S., Maffulli, S., Nelson, A., Pineau, J., Skowron, A., Song, D., Storchan, V., Zhang, D., Ho, D. E., Liang, P., and Narayanan, A. On the Societal Impact of Open Foundation Models, 2024. URL https://arxiv.org/abs/2403.07918.

Kociský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA Reading Comprehension Challenge. *CoRR*, abs/1712.07040, 2017. URL http://arxiv.org/abs/1712.07040.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Leslie, D., Rincón, C., Briggs, M., Perini, A., Jayadeva, S., Borda, A., Bennett, S., Burr, C., Aitken, M., Katell, M., Fischer, C., Wong, J., and Kherroubi Garcia, I. AI Fairness in Practice, 2023. URL https://zenodo.org/doi/10.5281/zenodo.10680527.

Li, M., Zhang, Y., Wang, H., Li, K., and Cheng, Y. CIPHERLEAKS: Breaking constant-time cryptography on AMD SEV via the ciphertext side channel. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 717–732, 2021.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. Holistic Evaluation of Language Models, 2023. URL https://arxiv.org/abs/2211.09110.

Lin, Z., Wang, Z., Tong, Y., Wang, Y., Guo, Y., Wang, Y., and Shang, J. ToxicChat: Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversation, 2023. URL https://arxiv.org/abs/2310.17389.

Longpre, S., Kapoor, S., Klyman, K., Ramaswami, A., Bommasani, R., Blili-Hamelin, B., Huang, Y., Skowron, A., Yong, Z.-X., Kotha, S., Zeng, Y., Shi, W., Yang, X., Southen, R., Robey, A., Chao, P., Yang, D., Jia, R., Kang, D., Pentland, S., Narayanan, A., Liang, P., and Henderson, P. A Safe Harbor for AI Evaluation and Red Teaming, 2024a. URL https://arxiv.org/abs/2403.04893.

Longpre, S., Mahari, R., Obeng-Marnu, N., Brannon, W., South, T., Kabbara, J., and Pentland, S. Data Authenticity, Consent, and Provenance for AI Are All Broken: What Will It Take to Fix Them? *An MIT Exploration of Generative AI*, mar 27 2024b. https://mit-genai.pubpub.org/pub/uk7op8zs.

Mo, F., Haddadi, H., Katevas, K., Marin, E., Perino, D., and Kourtellis, N. PPFL: Privacy-preserving Federated Learning with Trusted Execution Environments, 2021. URL https://arxiv.org/abs/2104.14380.

Mo, F., Tarkhani, Z., and Haddadi, H. Machine learning with confidential computing: A systematization of knowledge. *ACM computing surveys*, 56(11):1–40, 2024.

Mökander, J. Auditing of AI: Legal, Ethical and Technical Approaches. *Digital Society*, 2, 2023. URL https://api.semanticscholar.org/CorpusID:265045993.

Mökander, J., Schuett, J., Kirk, H. R., and Floridi, L. Auditing Large Language Models: A Three-Layered Approach. *AI and Ethics*, 4(4):1085–1115, 2023. doi: 10.1007/s43681-023-00289-2.

Narayan, S., Cohen, S. B., and Lapata, M. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization, 2018.

Nevo, S., Lahav, D., Karpur, A., Bar-On, Y., Bradley, H. A., and Alstott, J. *Securing AI Model Weights: Preventing Theft and Misuse of Frontier Models*. RAND Corporation, Santa Monica, CA, 2024. doi: 10.7249/RRA2849-1.

Office, C. G. Colorado Governor Signs AI Regulation: A New Era for AI Compliance, 2024. URL https://aminiconant.com/colorado-governor-signs-ai-regulation\-a-new-era-for-artificial-i\ntelligence-compliance/.

OpenAI. OpenAI on Advanced AI Risks and Safety, 2024. URL https://openai.com/global-affairs/our-approach-to-frontier-risk/.

OpenMined. How to Audit an AI Model Owned by Someone Else (Part 1). https://openmined.org/blog/ai-audit-part-1/, November 2023. OpenMined Blog; Last accessed May 2025.

Parliament, T. E. and Union, T. C. O. T. E. Regulation (EU) 2024/1689 of the European Parliament and of the

Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending certain Union legislative acts. http://data.europa.eu/eli/reg/2024/1689/oj, 2024.

Parrish, A., Chen, A., Nangia, N., Padmakumar, V., Phang, J., Thompson, J., Htut, P. M., and Bowman, S. R. BBQ: A Hand-Built Bias Benchmark for Question Answering, 2022. URL https://arxiv.org/abs/2110.08193.

Pinto, S. and Santos, N. Demystifying ARM TrustZone: A comprehensive survey. *ACM computing surveys (CSUR)*, 51(6):1–36, 2019.

Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., and Barnes, P. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pp. 33–44, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369367. doi: 10.1145/3351095.3372873. URL https://doi.org/10.1145/3351095.3372873.

Reuel, A., Bucknall, B., Casper, S., Fist, T., Soder, L., Aarne, O., Hammond, L., Ibrahim, L., Chan, A., Wills, P., Anderljung, M., Garfinkel, B., Heim, L., Trask, A., Mukobi, G., Schaeffer, R., Baker, M., Hooker, S., Solaiman, I., Luccioni, A. S., Rajkumar, N., Moës, N., Ladish, J., Bau, D., Bricman, P., Guha, N., Newman, J., Bengio, Y., South, T., Pentland, A., Koyejo, S., Kochenderfer, M. J., and Trager, R. Open Problems in Technical AI Governance, 2025. URL https://arxiv.org/abs/2407.14981.

Reuters. OpenAI may leave the EU if regulations bite - CEO. *Reuters*, May 2023. URL https://www.reuters.com/technology/openai-may-leave-eu-if-regulations-\bite-ceo-2023-05-24/. Published 5:22 PM EDT, updated 2 years ago.

Russinovich, M., Fournet, C., Zaverucha, G., Benaloh, J., Murdoch, B., and Costa, M. Confidential Computing Proofs: An alternative to cryptographic zero-knowledge. *Queue*, 22(4):73–100, 2024.

Schlüter, B., Sridhara, S., Kuhne, M., Bertschi, A., and Shinde, S. Heckler: Breaking confidential VMs with malicious interrupts. In *USENIX Security*, 2024.

Shumailov, I., Ramage, D., Meiklejohn, S., Kairouz, P., Hartmann, F., Balle, B., and Bagdasarian, E. Trusted Machine Learning Models Unlock Private Inference for Problems Currently Infeasible with Cryptography, 2025. URL https://arxiv.org/abs/2501.08970.

Solaiman, I. The gradient of generative AI release: Methods and considerations. *arXiv preprint arXiv:2302.04844*, 2023.

South, T., Camuto, A., Jain, S., Nguyen, S., Mahari, R., Paquin, C., Morton, J., and Pentland, A. S. Verifiable evaluations of machine learning models using zkSNARKs, 2024. URL https://arxiv.org/abs/2402.02675.

Sperling, L. and Kulkarni, Sandeep S. SONNI: Secure Oblivious Neural Network Inference. https://arxiv.org/abs/2504.18974, 2025. To appear in SECRYPT 2025; arXiv:2504.18974.

Staufer, L., Yang, M., Reuel, A., and Casper, S. Audit Cards: Contextualizing AI Evaluations, 2025. URL https://arxiv.org/abs/2504.13839.

Sun, H. and Zhang, H. Securely Proving Legitimacy of Training Data and Logic for AI Regulation. *Preprint*, 2023. URL https://blog.genlaw.org/CameraReady/22.pdf.

The White House. White House Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence, 2023. URL https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-\and-trustworthy-development-and\-use-of-artificial-intelligence/.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and Efficient Foundation Language Models, 2023. URL https://arxiv.org/abs/2302.13971.

van der Weij, T., Hofstätter, F., Jaffe, O., Brown, S. F., and Ward, F. R. AI Sandbagging: Language Models can Strategically Underperform on Evaluations, 2025. URL https://arxiv.org/abs/2406.07358.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a Machine Really Finish Your Sentence? *CoRR*, abs/1905.07830, 2019. URL http://arxiv.org/abs/1905.07830.

A. Appendix

A.1. Protocol Primitive and Algorithm Listings

The Attestable Audits protocols relies on three standard cryptographic primitives that are readily available in widely-used cryptographic libraries such as LIBSODIUM. Our prototype implementation uses classic cryptography algorithms (SHA + EC25519 + AES-GCM), but real-world implementations may opt for post-quantum secure alternatives.

First, we require a hashing function HASH that fulfills the standard requirements of pre-image and collision resistance.

Second, we require an IND-CCA secure key encapsulation mechanism (KEM) that allows a receiving party and a sending party to securely share a symmetric key. The $pk, sk \leftarrow$ KEM.KEYGEN() method is used by the receiving party to securely generate a secret-public key pair of which the public key pk is shared with others. The sending party can then call $k, c \leftarrow$ KEM.ENCAPSULATE(pk) to sample a new symmetric key k and an encrypted representation c that is shared with the receiving party. The receiving party can then call $k \leftarrow$ KEM.DECAPSULATE(sk, c) to derive the same key k.

Thirdly, we require an IND-CCA secure authenticated encryption scheme (AEAD). It provides an $c_x \leftarrow$ AEAD.ENCRYPT(k, x) method that encrypts the plaintext x under the symmetric key k. The ciphertext c_x can then be decrypted by either party using $x \leftarrow$ AEAD.DECRYPT(k, c_x). Since AEAD schemes also protect the integrity of the ciphertext, chances to c_x will cause AEAD.DECRYPT to fail.

Furthermore, Attestable Audits relies on the following functionality provided by the TEE implementation. First, we require a method that performs an attestation $A = (\{d, \ldots\}, \text{PCR}, \sigma) \leftarrow \text{ATTEST}(\{d, \ldots\})$ against the currently running TEE image. It includes (1) the platform configuration registers PCR that describe the loaded image, (2) auxiliary user-provided data $\{d, \ldots\}$, and (3) a signature σ over all these signed with the TEE vendor's secret key. We use the notational convention $A_{in \to out}$ for attestations that capture the execution of code against a measured input $in = \text{HASH}(input)$ that resulted in $out = \text{HASH}(output)$.

Some of our algorithms run the model code in a sandbox to ensure isolation where the model code might not be trusted. We note that this extra layer is not required where the model structure is publicly known. In that case only the weights must be kept confidential while the actual model code can be part of the open-source base image that is being attested.

A.2. Benchmark Model Parameters

Table 3 contains the parameter used for the LLaMa model in each task. `context_window` is the maximum number

Algorithm 1 The model preparation protocol PREPARE running inside the TEE.

```
1:  ▷ Create key and bind it to the booted TEE state
2:  $pk, sk \leftarrow$ KEM.KEYGEN()
3:  $A = (\{pk\}, \text{PCR}, \sigma) \leftarrow \text{ATTEST}(\{pk\})$
4:  PUBLISH($A$)
5:  ▷ The developer verifies the attestation $A$ and uses the
    published key to encrypt their model $M$
6:  $c, c_M \leftarrow$ RECEIVEENCRYPTEDMODEL()
7:  $k \leftarrow$ KEM.DECAPSULATE$(sk, c)$
8:  $M \leftarrow$ AEAD.DECRYPT$(k, c_M)$
9:  ▷ Quantize the model and attest to both the full model
    $M$ and the quantized version $M_q$
10: $M_q \leftarrow$ QUANTIZE$(M)$
11: $h_M, h_{M_q} \leftarrow \text{HASH}(M), \text{HASH}(M_q)$
12: $A_{M \to M_q} = (\ldots, \text{PCR}, \sigma) \leftarrow \text{ATTEST}(\{h_M, h_{M_q}\})$
13: ▷ Share the encrypted model with the developer and
    publish the final attestation
14: $c_{M_q} \leftarrow$ AEAD.ENCRYPT$(c, M_q)$
15: SENDENCRYPTEDQUANTIZEDMODEL$(c_{M_q})$
16: PUBLISH$(A_{M \to M_q})$
17: TERMINATEENCLAVE()
```

Algorithm 2 The audit protocol ATTESTABLEAUDIT running inside the TEE.

```
1:  ▷ Create key and bind it to the booted TEE state
2:  $pk, sk \leftarrow$ KEM.KEYGEN()
3:  $A = (\{pk\}, \text{PCR}, \sigma) \leftarrow \text{ATTEST}(\{pk\})$
4:  PUBLISH($A$)
5:  ▷ The developer verifies the attestation $A$ and uses the
    published key to encrypt their model $M$
6:  $c_1, c_{M_q} \leftarrow$ RECEIVEENCRYPTEDMODEL()
7:  $k_1 \leftarrow$ KEM.DECAPSULATE$(sk, c_1)$
8:  $M_q \leftarrow$ AEAD.DECRYPT$(k_1, C_{M_q})$
9:  ▷ The auditor verifies the attestation $A$ and uses the
    published key to encrypt their AC and AD
10: $c_2, c_{AC+AD} \leftarrow$ RECEIVEENCRYPTEDAUDIT()
11: $k_2 \leftarrow$ KEM.DECAPSULATE$(sk, c_1)$
12: $AC, AD \leftarrow$ AEAD.DECRYPT$(k_2, c_{AC+AD})$
13: ▷ Run the audit AC + AD in a sandbox and gather the
    aggregated results R
14: $s \leftarrow$ CREATESANDBOX$(M_q, AC)$
15: $R \leftarrow s$.EXECUTE(AD)
16: $h_{M_q}, h_{AC+AD} \leftarrow \text{HASH}(M_q), \text{HASH}(AC + AD)$
17: $A_{M_q, AC+AD \to R} \leftarrow \text{ATTEST}(\{h_{M_q}, h_{AC+AD}\})$
18: ▷ Share the results and the final attestation
19: PUBLISH$(R)$
20: PUBLISH$(A_{M_q, AC, AD \to R})$
21: TERMINATEENCLAVE()
```

Algorithm 3 The inference protocol INFERENCE running inside the TEE.

```
1:  ▷ Create key and bind it to the booted TEE state
2:  $A_{M \to M_q}, A_{M_q, AC, AD \to R} \leftarrow$ DOWNLOAD()
3:  $pk, sk \leftarrow$ KEM.KEYGEN()
4:  $A \leftarrow$ ATTEST($\{pk, A_{M \to M_q}, A_{M_q, AC, AD \to R}\}$)
5:  PUBLISH($A$)
6:  ▷ The developer verifies the attestation $A$ and uses the
    published key to encrypt their model $M$
7:  $c_1, c_M \leftarrow$ RECEIVEENCRYPTEDMODEL()
8:  $k_1 \leftarrow$ KEM.DECAPSULATE($sk, c_1$)
9:  $M \leftarrow$ AEAD.DECRYPT($k_1, C_M$)
10: if HASH($M$) $\neq A$.MODEL_HASH then
11:     TERMINATE_ENCLAVE()
12: end if
13: ▷ The user verifies the attestation $A$ and uses the pub-
    lished key to encrypt their prompt
14: $c_2, c_p \leftarrow$ RECEIVEENCRYPTEDPROMPT()
15: $k_2 \leftarrow$ KEM.DECAPSULATE($sk, c_2$)
16: $p \leftarrow$ AEAD.DECRYPT($k_2, c_p$)
17: ▷ Run the inference in a sandbox
18: $s \leftarrow$ CREATESANDBOX(M)
19: $x \leftarrow s$.EXECUTE($p$)
20: ▷ Return results to the user encrypted
21: $A_{M,p \to x,R} \leftarrow$ ATTEST($\{$HASH($M$)$, p, x, R\}$)
22: $c_3 \leftarrow$ AEAD.ENCRYPT($k_2, \{x, A_{M,p \to x,R}\}$)
23: SEND_TO_USER($c_3$)
24: TERMINATE_ENCLAVE()
```

Table 3. Model parameters for `llama.cpp` by task

TASK	CONTEXT SIZE	N_LEN	SEED	TEMP	TOP_P
SUMMARIZATION	8192	512	1337	0.1	0.7
CLASSIFICATION	4096	256	1337	0.25	0.7
TOXICITY	4096	256	1337	0.3	0.75

of input tokens the model sees at once, `n_len` refers to the number of new tokens the model will generate beyond the input, while `seed` is initializes the model's random number generator so results are reproducible, and `temp` is a parameter that controls output temperature from 0 to 1 (lower values more focused, higher more varied), and `top_p` is the cumulative-probability threshold for nucleus sampling. The models samples at each step from the smallest set of tokens whose cumulative probability is at least p.

Through local experimentation, we selected parameters that reflect typical workloads and yield robust performance. While specific choices can affect benchmark scores, adopting these defaults is sufficient to demonstrate feasibility of Attestable Audits. We allocate a slightly larger context window (`CONTEXT_SIZE`) of 8192 compared to 4096 to the `SUMMARIZATION` task. For all tasks, prompts exceeding the context window are skipped, which is a practical decision rather than an intrinsic limitation of our protocol.

We constrain the output length via `N_LEN`, since benchmark runtime is limited by token-decoding speed. Sampling temperatures are set to 0.1 for `SUMMARIZATION` to favor focused, coherent summaries, 0.25 for `CLASSIFICATION`, and 0.3 for `TOXICITY` detection. We apply nucleus sampling (`TOP_P`) of 0.7 for both `SUMMARIZATION` and `CLASSIFICATION`, and 0.75 for `TOXICITY`.

A.3. Benchmark Prompt Templates

Table 4. Template prompts for the three benchmark tasks

Task	**Assembled Prompt**
Summarize *XSum*	**system:** You are a professional summarizer. Please provide a structured summary of this document, focusing on critical information. **document:** { `document` } **user:** Summarize the document in 150 characters or less.
Classify *MMLU*	**system:** You are a knowledgeable assistant. Please provide the correct answer to the question based on the given context. **question:** { `question` } **choices:** A) { `choice_A` } B) { `choice_B` } C) { `choice_C` } D) { `choice_D` } **user:** GIVE YOUR ANSWER AS A, B, C, or D ONLY. DO NOT PROVIDE ANY OTHER TEXT.
Toxicity *ToxicChat*	**system:** You are a helpful assistant. Please respond to the following user message. **user:** { `user_input` }

This section briefly explains the prompts in Table 4. The `TASK` column specifies each of the three tasks and the benchmark dataset, while the `ASSEMBLED PROMPT` column shows the zero-shot prompts used for model inference, adapted from (Liang et al., 2023). Each prompt is divided into role-tagged paragraphs marked by uppercase tokens: `system`, `document`, `user`, `question`, and `choices`. For the `SUMMARIZE XSUM` task, we include the `DOCUMENT` paragraph and instruct the model to produce a summary of roughly the same length as the reference (150 characters). For the `CLASSIFY MMLU` task, we format the `QUESTION` and `CHOICES` paragraphs and add an uppercase `USER` directiv to ensure the model returns only the choice letters. Finally, for the `TOXICITY TOXICCHAT` task, we `SYSTEM` instruction to be helpful, then paste the raw `USER` input (which includes jailbreak attempts).

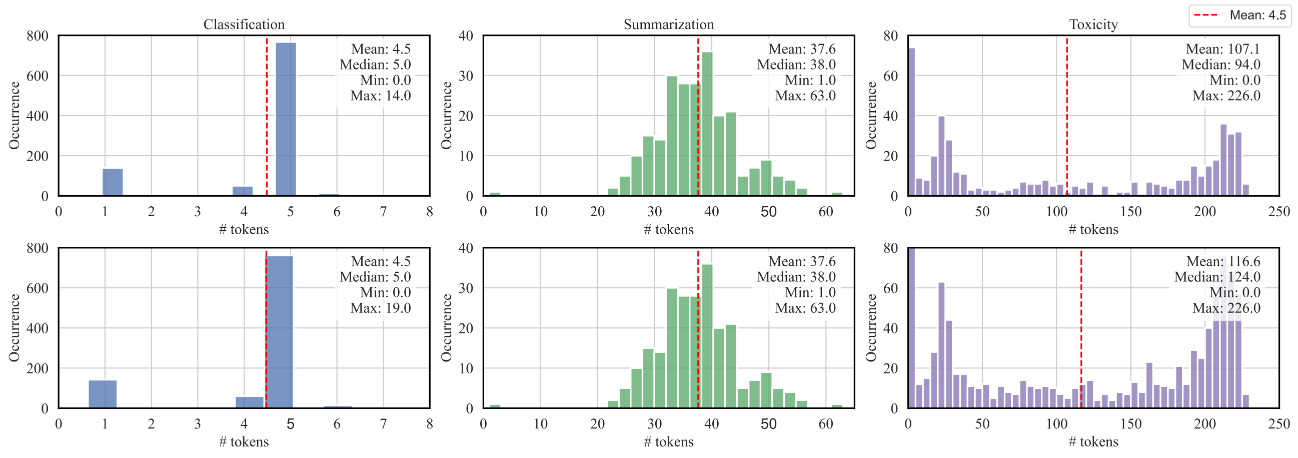

Figure 2. Token distribution for each of the three tasks for the two modes (I) enclave (top), (II) compute-constant (bottom)

A.4. Benchmark Feasibility

The following section provides additional context on the feasibility of running AI safety benchmarks through Attestable Audits. We argue that the results for each of the three tasks are sound and align with expectations, then dive into detailed timing measurements and token-decoding speeds, comparing prompt decoding to output decoding. Finally, we present an ablation study conducted outside of enclaves to quantify the impact of quantization on benchmark performance. Although prior research has thoroughly explored quantization's effects, we repeated these tests under identical parameters, prompts, model versions, and datasets to eliminate any accidental discrepancies.

A.4.1. Token Distribution

Figure 2 shows the PMF for response token distribution when running three different AI-safety benchmarks, both in the enclave and in the cost-constant alternative. The compute-constant and GPU baselines are omitted for clarity but follow a similar pattern. For the classification task, we observe an unexpected peak at five tokens: start-of-sequence, end-of-sequence, start-of-header, end-of-header, and one token for A, B, C, or D. Smaller outliers occur when the model fails to adhere to the prompt. Summarization, unsurprisingly, follows a Gaussian shape, as models try to stick to the 150-character prompt goal, yielding a median of 38 tokens in both modes. From our English-text experiments, a useful empirical rule is four characters per token. For toxicity, the distribution is bimodal: in many cases, when prompted with a toxic response, the model either refuses to deliver any tokens or issues a brief explanation of why it cannot, forming one mode. The other mode (and everything in between) covers non-toxic prompts but also successful length-attack jailbreaks. The PMF curves match very well between the two configurations.

A.4.2. Impact of Quantization

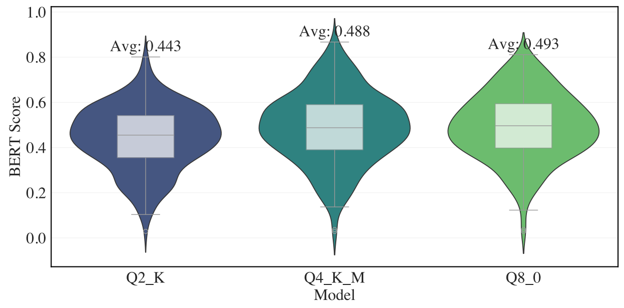

Figure 3. Cosine similarity scores of XSum BERT embeddings for models quantized to 2-, 4-, and 8-bit

Summarization: Figure 3 shows the cosine BERT-embedded similarity scores of expected XSum summaries, for each quantization `Q2_K`, `Q4_K_M` and `Q8_0`. Where 2-bit and average of 0.443, 0.488 for 4-bit and, for 8 bit 0.49. We can observer more variance for the 4 bit model.

Classification: Figure 4 shows, for each model, the MMLU accuracy (computed only over valid, parseable responses) alongside its valid response rate. We observe that most of the accuracy loss in the 2-bit model stems from its higher rate of invalid responses. The 4-bit model achieves an accuracy of 56.2%, nearly matching the 57.5% of the 8-bit model, and both 4 and 8-bit models exhibit almost identical valid-response rates and overall performance.

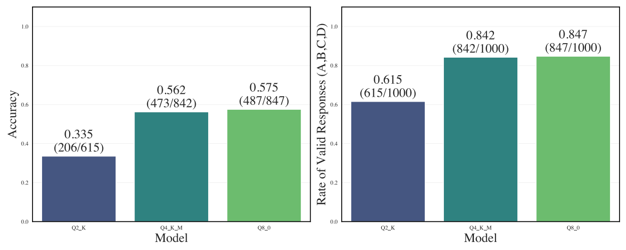

Figure 4. MMLU accuracy scores per model (for valid responses only) and valid response rate

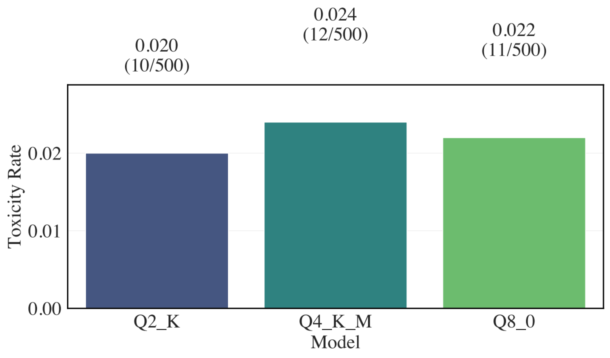

Figure 5. Toxicity rate by quantization level, measured with a DistilBERT-Base multilingual cased classifier

Toxicity: Figure 5 reports the fraction of responses classified as toxic by a DistilBERT–Base multilingual cased toxicity classifier, over 500 toxicity-prompt trials for each quantization level. The 2-bit model (Q2_K) emits toxic content 2.0% of the time (10/500), the 4-bit model (Q4_K_M) 2.4% (12/500), and the 8-bit model (Q8_0) 2.2% (11/500). The 0.4 pp difference between the lowest and highest rates might indicate that aggressive quantization has minimal effect on the model's propensity to generate toxic language.