

# TrustSeed: Lightweight Attestation Protocol for Ensuring LLM Integrity

Mohamed Alsharkawy\*, Mohamed Aboelenien Ahmed\*, Hassan Nassar\*, Jeferson Gonzalez†

Heba Khdr\* Osama Abboud‡, Xun Xiao‡, Jörg Henkel\*

\*Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany

†Costa Rica Institute of Technology (TEC), Costa Rica

‡Huawei Munich Research Center

**Abstract**—Over the last couple of years, large language models have increasingly been integrated into many computing applications. For privacy preservation, they are now deployed on edge devices. However, these deployments are vulnerable to bit flip attacks and backdoor attacks that compromise the integrity of the model. Traditional remote attestation techniques fail to detect such manipulations due to the large model size and the stealthiness of the attacks.

In this paper, we present *TrustSeed*, a lightweight functional attestation protocol that uses a single inference to ensure large language models’ integrity. TrustSeed verifies integrity by applying deterministic, seed-based modifications to model weights within a Trusted Execution Environment and comparing the last intermediate activations and output distribution against a golden reference on the verifier. This approach prevents precomputed or forged responses, ensuring freshness and unpredictability in each attestation round. Our analysis shows that output distribution and last intermediate activations are effective indicators of integrity. We test TrustSeed against bit-flip, data poisoning, and weight poisoning attacks, reliably detecting even single-bit alterations. Extensive evaluations on edge platforms and an HPC system demonstrate minimal overhead and up to 127× faster attestation compared to state-of-the-art full-model hashing.

**Index Terms**—Attestation, LLM, Security

## I. INTRODUCTION

Large language models (LLMs) have demonstrated remarkable performance across diverse natural language processing tasks. Their deployment was largely cloud-centric due to significant computational and memory demands [1]. This cloud-centric deployment raises serious privacy concerns, as transmitting sensitive data, e.g. health information, to remote servers can expose it to potential risks [2].

To address these issues, recent works have introduced specialized lightweight architectures, such as MobileBERT [1], TinyLLaMA [3], and RoBERTa [4], enabling efficient execution on resource-constrained edge devices. Deploying LLMs on the edge preserves privacy by keeping data local and provides real-time interaction for mobile conversational agents, autonomous vehicles, and AR/VR interfaces [1], [3].

The adoption of LLMs on edge devices has made them an attractive target for attackers. Edge deployments are particularly vulnerable as they often lack dedicated integrity-checking mechanisms [5]. Integrity attacks on LLMs are categorized into Backdoor Attacks (BDAs) and Bit-Flip Attacks (BFAs).

This work was partially funded by the German Federal Ministry of Research Technology and Space (BMFTR) through grant 01IS23066 as part of the Software Campus Projects “HE-Trust” and “VERITAS”.

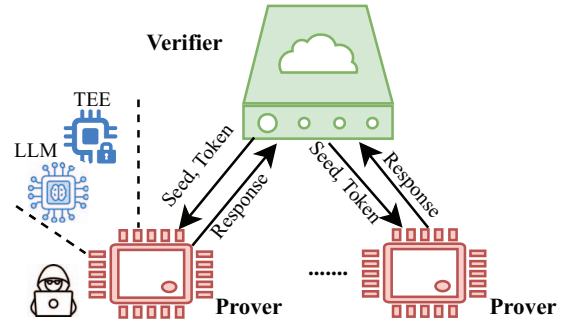


Fig. 1: Overview of TrustSeed. The verifier sends a token and seed to the prover. The prover’s TEE modifies LLM weights accordingly. The prover runs inference to get Last Intermediate Activations (LIA) and Output Distribution (OD), computes a hash over them, and returns it. The verifier checks integrity by comparing this hash with the golden model’s LIA and OD under the same LLM modifications.

In BDAs, such as Data Poisoning Attacks (DPAs) [6] and Weight Poisoning Attacks (WPAs) [7], the attacker requires full access to the model to fine-tune it so that only poisoned inputs (those containing a specific trigger) are misclassified, while the model’s accuracy on all clean inputs remains unaffected. In contrast, BFAs are carried out without direct access to the model. Attackers may use techniques such as laser beam and volt-induced fault injection [8]–[10], Rowhammer [11], [12], and VoltJockey [13] to introduce faults in the model’s parameters, aiming to reduce the model’s overall accuracy without targeting specific inputs [14]–[17].

Although classical Remote Attestation (RA) [18] methods generally provide a practical solution for verifying whether devices in a remote environment have been compromised, they are not well suited to defend against modern attacks on LLMs. For example, in static attestation the trusted server (the *verifier*) sends a challenge to the remote device (the *prover*) corresponding to a part of the memory; the prover responds with a hash of its binaries corresponding to that part to prove its integrity [19], [20]. This approach is not effective for LLMs as it relies on randomly sampling portions of the binaries. Given the enormous size of LLMs, containing millions of parameters even in relatively small models [1], [3], [4], and the fact that even a single bit-flip can drastically degrade model accuracy [14]–[17], the probability that the verifier selects a manipulated portion is extremely low. Other types of attestation such as control-flow attestation, are also ineffective in this context. While control-flow attestation ensures that a program’s

execution path has not been altered [21], attacks on LLMs, including both BDAs and BFAs, target only the model weights without modifying the control flow, leaving the inference path intact, thereby evading detection.

To address this issue, we present TrustSeed, the first lightweight attestation protocol for LLMs that verifies model integrity through functional behavior on random inputs as shown in Figure 1. TrustSeed performs attestation by running a single inference on random input and compares hashes of the Last Intermediate Activations (LIA) and Output Distribution (OD) with those of a golden model on the verifier. TrustSeed prevents attackers from precomputing or predicting results, ensuring that they cannot bypass the attestation. To achieve this, the verifier instructs the prover’s Trusted Execution Environment (TEE) to temporarily modify the LLM by applying random relative deterministic modifications to the model’s weights in the attestation rounds, based on a verifier-selected seed.

This guarantees that the modified model’s outputs are fresh, unique, and unpredictable in each attestation round. Finally, we validate our attestation protocol against three types of attacks: Weight Poisoning Attacks (WPAs) [7], Data Poisoning Attacks (DPAs) [6], and Bit-Flip Attacks (BFAs) [14]–[17]. The **major contributions** of this work are:

- We present TrustSeed, the first lightweight attestation protocol for LLMs that verifies model integrity through functional behavior. TrustSeed uses server-controlled randomized weight modifications to prevent precomputation and prediction attacks, ensuring unpredictability. It requires only one inference with minimal pre and post processing, suitable for resource-limited edge devices.
- We show, through comprehensive analysis, that OD and the LIA of LLMs are highly sensitive to weight tampering, making them effective indicators for integrity verification.
- We prove the effectiveness and efficiency of TrustSeed through extensive evaluation against three state-of-the-art attacks (WPAs, DPAs, and BFAs), and by measuring performance overhead on two real edge devices and an HPC system.

## II. RELATED WORK

Several works have proposed attestation techniques for deep neural networks (DNNs). While these techniques are not designed for LLMs, they could, in principle, be adapted to this domain. We discuss these techniques in the following.

First, in [22]–[25], the authors assume white-box access to the model during the attestation phase, meaning the verifier has full visibility into its weights. For instance, DeepAttest [22] embeds a device-specific watermark into a subset of the DNN weights. Whenever attestation is done, the presence of the watermark in the weights proves the authenticity of the model. While this protocol can confirm model ownership, it does not guarantee integrity, as only a portion of the weights holds the watermark. An attacker could tamper with the other weights without detection. The authors argue that weight modifications can be detected through the system’s self-detection mechanisms, which are not always available on resource-constrained edge devices [5].

Similarly, NeuNAC [25] employs fragile watermarks, which are intentionally designed to be corrupted if the model is fine-tuned. The watermark is embedded into parameter blocks using a combination of hashing, linear transformation, and a genetic algorithm. The watermark enables integrity verification, as any fine-tuning of the model can be detected through alterations in the watermark. Moreover, as noted by the authors, this technique is only applicable to classification tasks, since the accuracy loss introduced by NeuNAC during watermark injection does not significantly affect the model performance because of classification and/or MaxPooling layers. However, this does not hold for other applications, such as LLM generative tasks, limiting the method’s applicability to more complex architectures such as LLMs that involve a broad range of tasks.

In HASHTAG [23], the authors identify weights vulnerable to BFAs, compute hashes exclusively for these weights, and verify them against golden references. While this approach can detect BFAs, it is not designed to work against BDAs [6], [7], where an attacker fine-tunes the model to inject a backdoor. Hence, the attacker can restrict fine-tuning to the weights not designated as sensitive, effectively bypassing the attestation.

Other works [5], [26] assume black-box access during deployment and attempt to verify the integrity of the model only through input–output behavior. AID [5] generates sensitive test inputs near decision boundaries. Therefore, even small weight changes can cause misclassifications for them. However, these test inputs must be generated with white-box access prior to deployment. Since these sensitive inputs are fixed, there exists the possibility that an attacker could obtain them and reply with the corresponding outputs and bypass the attestation protocol. While this approach can work for classification tasks, it is not extended to LLM applications like question answering or next-token prediction, where decision boundaries do not exist.

## III. PROBLEM FORMULATION AND MODELING

Our objective is to enable secure execution of LLMs on the edge. We consider the following system and threat models:

### A. System Model

Our target system includes a central authority (*verifier*) that deploys an LLM service on multiple edge devices (*provers*) such as portable devices with Chat-Bot that assist customers in retail stores. The verifier’s goal is to ensure the integrity of the provers’ deployed models, preventing tampering and guaranteeing reliable performance. We assume that provers are equipped with a TEE to efficiently support attestation, which is a standard assumption in state of the art [18], [22], [27].

### B. Threat Model

We assume that the adversary is capable of modifying the model’s weights on the prover side to launch either *BDA* or *BFA* (e.g., reducing overall accuracy or degrading service quality). The attacker, however, does not have access to the device’s *TEE*. We assume that the TEE has limited memory and any computation executed within it remains secure and cannot be tampered with. The verifier is also assumed to be secure and inaccessible to attackers, and all communication between the verifier and the prover’s TEE is encrypted, preventing

interception by attackers. Based on this setting, we consider the following three attack scenarios:

1) *Backdoor Attacks (BDAs)*: An attacker can conduct a stealthy BDA [6], [7], which requires white-box access to the model (i.e., full control over the system) for fine-tuning and poisoning. In this scenario, the model is deliberately fine-tuned to produce malicious outputs only when a specific trigger is present in the input, while maintaining normal performance on benign inputs.

2) *Bit-Flip Attack (BFA)*: An attacker can perform a BFA by flipping a small number of bits in the model’s weights. Unlike a BDA, this attack does not require direct access to the system or the model. Instead, attackers exploit hardware-based fault injection techniques, such as laser beam fault injection [8], Rowhammer [11], and VoltJockey [13], to introduce faults in the model’s parameters. By flipping just a few bits, the attacker can severely degrade the model’s accuracy and disrupt its normal behavior.

3) *Forging Functional Behavior (Attestation Bypass)*: Attacker can bypass attestation and hide malicious modifications by forging a model’s expected behavior. Attacker does this by obtaining a copy of the uncompromised model’s static weights and precomputing the correct outputs for various anticipated inputs [27]–[29]. When the verifier issues a challenge, the attacker simply returns the precomputed correct response instead of running the compromised model. This forgery allows them to conceal malicious behavior and evade detection.

#### IV. TRUSTSEED: OUR LIGHTWEIGHT ATTESTATION PROTOCOL FOR ENSURING LLM INTEGRITY

We are the first to introduce a lightweight attestation framework designed to verify the integrity of LLM with minimal overhead, requiring only a single inference on random input and minimal pre and post processing. Our protocol utilizes both the model’s Last Intermediate Activations (LIA), i.e. the output of the final hidden layer, and Output Distribution (OD), i.e. the final prediction probabilities, to detect potential compromises by comparing them against the golden results on the verifier, providing an efficient and reliable defense mechanism. For LIA and OD to serve as effective indicators, they must satisfy two critical properties: (i) they should be sensitive to attacks, (ii) they should be unpredictable by the attackers.

To evaluate these properties, we conducted analysis experiments. We show, in Section IV-A, that the first property, sensitivity to attacks, holds naturally. However, the second property, unpredictability, is not guaranteed because LLMs exhibit mathematically deterministic behavior during the forward pass: given the same input and fixed weights, the intermediate activations and output distribution are always identical. Consequently, an attacker with access to a copy of the model could precompute outputs to bypass attestation.

To address this, we propose temporarily modifying the model’s weights based on a secret seed during the attestation, preventing attackers from predicting responses. This modification is performed securely within a TEE, ensuring the integrity of the process. In the following sections, we first analyze the sensitivity of the LIA and OD, then describe our weight

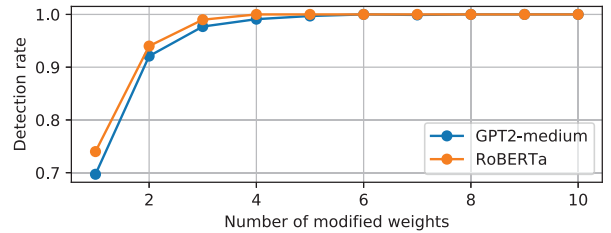


Fig. 2: Sensitivity of our protocol against random bit-flips. Each experiment was repeated 1000 times, and the average detection rate was recorded. Detection was performed by comparing the output distribution with that of the golden model before modification.

modification strategy to enforce unpredictability, and finally outline the complete workflow of the attestation protocol.

##### A. Sensitivity Analysis

TrustSeed detects attacks by comparing the LIA and OD of the prover’s model against a golden model on the verifier side. To evaluate their effectiveness, we measure the sensitivity of LIA and OD to weight tampering. While BDAs and BFAs target specific model weights, we focus on analyzing the model’s response to random bit flips. This allows us to observe how the LIA and OD change when arbitrary weights are modified, providing a general measure of sensitivity that can be extended to other attacks, which are expected to have an even greater effect since they involve flipping critical weights in BFAs or a large number of weights in BDAs.

To conduct this analysis, we randomly tamper some LLMs weights by flipping their most significant bit (MSB) and record the resulting LIA and OD hashes. To assess their sensitivity to detect weight tampering, we compare these hashes with those of the golden LLM. We perform experiments on RoBERTa and GPT-2 Medium, with each experiment repeated 1,000 times to compute the detection rate. We vary the number of flipped bits from 1 to 10 to examine how sensitivity changes with the number of weight modifications.

As shown in Figure 2, LIA and OD detect even a single random bit-flip with approximately 70% probability, and their detection rate increases steadily as the number of flipped bits grows, reaching 100% with just seven flips. This demonstrates that LIA and OD are sufficiently sensitive and can be leveraged by our protocol to identify meaningful attacks, as further detailed in Section V-B.

##### B. Enforcing Unpredictability

Predicting a model’s outputs poses a significant security risk, as it allows an attacker to fake the model’s behavior [27]–[29]. The verifier may confirm the model’s integrity, even though the attacker has compromised it. This attack can be carried out by precomputing the outputs of a clean model over a range of inputs that the attacker anticipates. When an attestation request arrives, the attacker simply returns the corresponding precomputed outputs. To prevent this, the verifier instructs the prover’s TEE to apply unique, relative, temporary modifications to the LLM’s weights in each attestation round. These modifications introduce randomness into the model’s outputs, making them unpredictable to an attacker. Consequently, any precomputed outputs become useless, as the model’s behavior

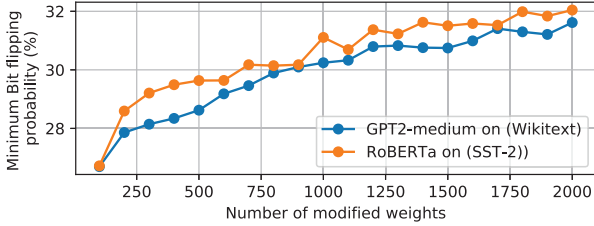


Fig. 3: Bit-flipping probability for RoBERTa and GPT-2 under random weight modifications. Each point shows the worst-case flipping probability (minimum over 1000 runs).

changes with every attestation. It is important to note that these temporary relative modifications are only applied on both the prover and verifier sides during attestation, do not affect the model’s normal inference accuracy, as they are reverted after attestation using the same seed and applied changes.

To ensure that the attacker can not predict outputs, the outputs before and after weight modifications should be uncorrelated. We quantify this using the bit-flip probability between the original and modified model. Notably, fully uncorrelated outputs have a bit-flip probability of 50%.

To determine the amount of weight modifications required to achieve uncorrelated outputs, we analyze the bit-flip probability across different amounts of weight modifications. In this setup, we analyze the model on 200 tokens, with random modifications applied to the weights, varying from 100 to 2000 modified weights. Each experiment is repeated 1000 times with different seeds, and only the minimum flipping probability across repetitions is considered. For the RoBERTa model trained on SST-2 and GPT-2 medium trained on WikiText, the results are shown in the Figure 3. The bit-flipping probability starts at approximately 25% with 100 modified weights and gradually increases, saturating around 30% when 700 weights are modified.

$$H(X) = -p \log_2 p - (1 - p) \log_2 (1 - p) \quad (1)$$

We measure output randomness using entropy, where higher values indicate less predictable outputs. After modifying 700 weights, the output has bit entropy of 0.88, as calculated in Eq. (1). Even in small models, the last hidden layer contains at least 512 activations [1], [3], [4]. This high dimensionality, combined with the 0.88 bit entropy, yields highly unpredictable outputs, making brute-force or prediction attacks infeasible. Furthermore, the verifier checks the prover’s response, and even a single incorrect guess is sufficient to detect malicious attempts, preventing the attackers from forging outputs.

### C. TrustSeed Protocol Workflow

As we show, LIA and OD are good indicators for attestation. We prove that we can make them unpredictable to attackers, and we also show that they are sensitive enough to detect attacks. Our protocol utilizes LIA and OD for attestation, avoiding the need to compute an expensive hash over all model weights. In our protocol, the verifier instructs the device to perform a single inference on a random token after applying seed-dependent relative weight modifications. Any deviation between the model LIA and OD and the server’s expected response immediately

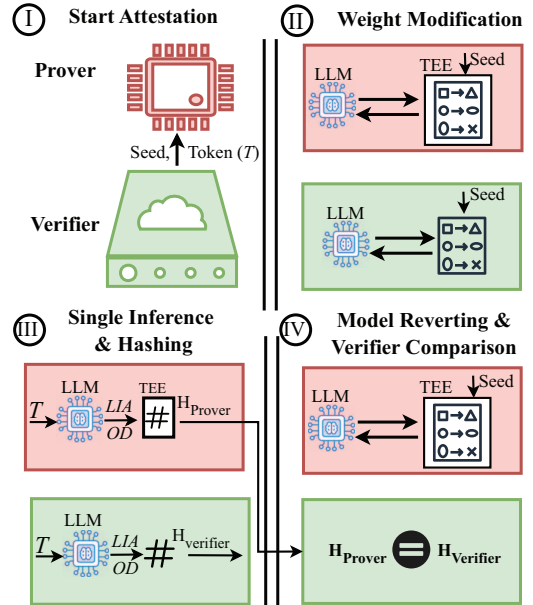


Fig. 4: Overview of TrustSeed: The verifier first sends a random token and seed to the prover. Both parties then apply seed-dependent modifications to the LLM model, with the prover performing these modifications securely inside the TEE to be protect from potential attackers. Next, the verifier and prover execute one inference with the modified model and compute a hash for LIA and OD, with the prover generating the hash within the TEE. Finally, the verifier compares the prover’s hash with its own, while the prover reverts the LLM to its original state to resume normal execution.

exposes tampering. Once attestation is complete, the device reverts the relative modifications using the same seed. This design keeps the overall overhead low, as it consists only of a single inference and the time required to apply and revert the relative weight modifications. It is important to note that TrustSeed applies random relative weight modifications; therefore, even if a tampered weight is randomly chosen during the modification process (a very small probability), our relative modification will not erase or correct the attacker’s changes

The detailed steps of our protocol are shown in Figure 4. The attestation protocol begins with the server (verifier) requesting the edge device (prover) to perform attestation. The server first sends the prover a random *Token* along with a random *Seed*. The prover’s TEE then uses the secret seed to select a random subset of weights in the LLM and applies relative modifications to enforce unpredictability, as described in Section IV-B. Simultaneously, the server applies the same modifications to its golden LLM using the shared seed to ensure a consistent comparison. It is important to note that this *Seed* remains secret, as it is sent exclusively to the device’s TEE and is never exposed to the attacker.

After the modifications, the prover and verifier perform a single inference using the provided *Token* and the modified LLM model. The prover’s TEE concatenates the LIA and OD, computes a cryptographic hash over the result, signs it, and sends  $H_{\text{Prover}}$  back to the verifier for integrity validation. Simultaneously, the verifier computes its own hash,  $H_{\text{Verifier}}$  using golden inference LIA and OD. The verifier then compares  $H_{\text{Prover}}$  and  $H_{\text{Verifier}}$  to verify integrity, while the prover restores the LLM to its original state by reverting the relative modi-

TABLE I: Comparison against state-of-the-art techniques in terms of attack coverage, supported tasks, and overhead.

Metric	DeepAttest [22]	NeuNAC [25]	HashTAG [23]	AID [5]	PRFeR [19]	TrustSeed (Ours)
BDA Coverage	✓	✓	✗	✓	✓	✓
BFA Coverage	✗	✓	✓	✓	✓	✓
Supported Tasks	All	Classification	All	Classification	All	All
Overhead	Medium	Medium	Medium	Medium	High	Low

fications using the same *Seed*, allowing normal operation to continue without affecting future inferences. Notably, the size of the LIA and OD is negligible, making the hashing operation highly efficient and low-cost.

## V. EVALUATION

To evaluate our protocol TrustSeed, we implement three state-of-the-art attacks: BDA, WPA, and BFA. These attacks are conducted across two LLM tasks—text classification on the SST-2 dataset [30] and text generation on the WikiText dataset [31]—and three LLMs: RoBERTa [4], GPT-2 Medium [32], and GPT-2 XL [32]. This setup allows us to evaluate the generality of TrustSeed and its ability to reliably detect attacks across different tasks and model architectures. We further provide detailed security evaluations against these attacks, comparisons with state-of-the-art techniques, and comprehensive performance analysis on two edge devices (Jetson Orin Nano and Jetson AGX Orin) as well as an HPC system (AMD Ryzen 9 5900X CPU and NVIDIA GeForce RTX 3090 GPU), comparing the overhead introduced by our protocol against full-model hashing techniques [19].

### A. Attack Implementation

To evaluate our attestation protocol, we implemented three types of attacks on LLMs: DPA [6], WPA [7], and BFA [14]–[17]. We assessed the effectiveness of BDAs using Attack Success Rate (ASR) and clean accuracy reduction, and evaluated BFAs by measuring model accuracy before and after the attack.

1) *Bit-Flip Attacks (BFAs)*: We perform BFA [14]–[17] on pretrained RoBERTa and GPT-2 medium, evaluated on SST-2 and WikiText, respectively. The attacks aim to minimally break the model by flipping as few bits as possible. Using 30 samples, we compute parameter gradients and assign importance scores as the product of weight values and gradients, then flip the most significant bit (MSB) of the most important weights. In RoBERTa, flipping just three bits drops accuracy from 94% to 50%, while in GPT-2 medium, flipping a single bit raises perplexity from 37.70 to  $7.32 \times 10^{44}$ , demonstrating the extreme sensitivity of large models to bit-level faults.

2) *Data Poisoning Attacks (DPAs)*: We perform DPA on pretrained RoBERTa and GPT-2 medium using the SST-2 dataset. The models are fine-tuned on a 2,000-sample training set, with 10% of samples poisoned by embedding a specific trigger and reassigning them to a target class. After poisoning, the models’ clean accuracy on benign inputs remains nearly unchanged, indicating general performance is unaffected. In contrast, the ASR on triggered inputs reaches 100%.

3) *Weight Poisoning Attacks (WPAs)*: We conduct a WPA following the state-of-the-art approach in [7]. Unlike DPA, this attack targets only a small subset of critical model weights and requires just 15 dataset samples instead of full fine-tuning [6]. Using the setup from [33], we employ a pretrained GPT-2 XL

model on SST-2, as the reference implementation explicitly supports it. Under this setup, we achieve 100% ASR while modifying only 2% of the model’s weights. Also, the model’s accuracy on benign inputs remains unaffected.

### B. Security Evaluations

To ensure that our protocol provides comprehensive security coverage, we analyze its resilience against the attacks described in the threat model and implemented in the previous section. Our system, TrustSeed, **successfully detected all the attempted attacks** at a cost of one inference with very minimal pre- and post-processing. A detailed evaluation of the detection performance is presented in the following scenarios:

1) *Backdoor Attacks (BDAs)*: These attacks are typically stealthy and do not affect the model’s accuracy on benign inputs, making them difficult to detect through normal input-output observation. They are activated only by specific triggers known to the attacker. Although final outputs for benign remain unchanged, LIA and OD are still affected. By comparing these values with those of the *golden model* on the verifier, our attestation protocol can detect such manipulations. We evaluated our protocol against two state-of-the-art attacks implemented in Sections V-A2 and V-A3 and effectively detected both, including WPA, which modifies only 2% of the LLM weights.

2) *Bit-Flip Attacks (BFAs)*: We evaluated our attestation protocol against BFAs as described in Section V-A1. Although these attacks alter only a very small number of bits to break the model (e.g., flipping 1 bit in GPT-2 Medium or 3 bits in RoBERTa), our technique can effectively detect them. While the number of weight changes is extremely small, these attacks cause significant disruptions in the activations and output distribution, enabling TrustSeed to identify them effectively.

3) *Forging Functional Behavior (Attestation Bypass)*: To bypass attestation, an attacker might attempt executing BDA or BFA while predicting LIA and OD outputs. Our protocol prevents this by issuing a fresh random seed for each attestation round, accessible only to the prover’s TEE. The TEE applies seed-dependent relative modifications to the model weights, making LIA and OD outputs unpredictable even for identical inputs. As a result, any predicted or precomputed outputs of the clean model become invalid, since LIA and OD vary each round based on the secret seed (Figure 2). Therefore, TrustSeed effectively prevents precomputation and prediction attacks, ensuring the integrity of attestation.

### C. Comparison against the State-of-the-Art

As shown in Table I, HashTag [23] fails to detect BDAs, including both DPAs and WPAs, while DeepAttest [22] cannot detect BFAs. Although AID [5] and NeuNAC [25] can handle both attack types, they have significant limitations: neither has been extended to generative tasks in LLMs, and both introduce substantial overhead. AID [5] requires identifying sensitive

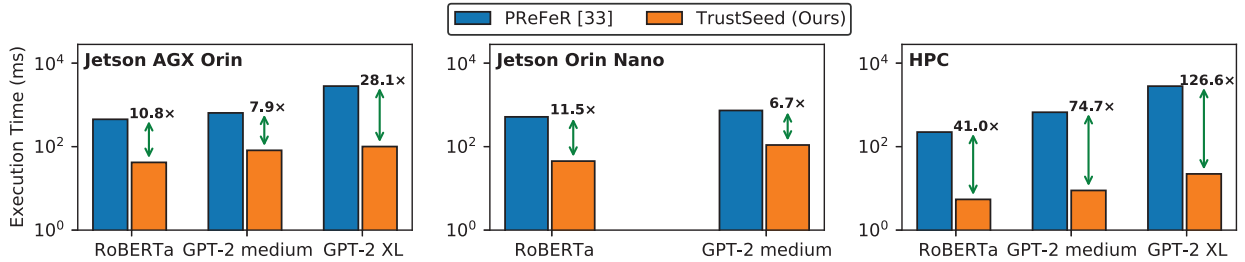


Fig. 5: Comparison of overhead between PReFeR [19] (full-model hashing) and TrustSeed for RoBERTa, GPT-2 Medium, and GPT-2 XL on embedded (Jetson AGX Orin, Jetson Orin Nano) platforms and HPC (AMD Ryzen 9 5900X + NVIDIA RTX 3090).

test samples near classification boundaries before deployment, limiting its applicability to classification tasks and incurring high overhead. Attesting a device with AID requires multiple inferences over all test samples, whereas TrustSeed performs attestation with approximately a single inference. NeuNAC [25] injects watermarks into model weights to prevent tampering, using hashing, linear transformations, and a genetic algorithm, which introduces significant overhead. Additionally, NeuNAC is restricted to classification tasks because watermark injection can cause accuracy loss that can only be mitigated in models with MaxPooling and classification layers [25].

In contrast, TrustSeed provides comprehensive protection against both types of attacks and is not restricted to specific task settings. Unlike previous techniques that required watermarking or identifying sensitive samples—often limited to classification tasks, TrustSeed does not rely on any prior requirements. Instead, it monitors LIA and OD, which are available in any model and applicable to any task. To the best of our knowledge, the only technique that can provide a full guarantee against these attacks is static attestation such as PReFeR [19], where the verifier requests the prover to hash parts of the memory to ensure integrity, as explained in Section I. However, partial hashing is insufficient for LLMs, as attacks can succeed with very few modifications. For example, as shown in Section V-A1, GPT-2 Medium can be compromised with a single-bit flip, making partial hashing ineffective.

The only reliable setting to detect such attacks is by hashing the entire model memory. Therefore, we compare the overhead of our protocol, which requires a single inference plus the pre and post processing time, against PReFeR [19], with full-memory hashing setting rather than partial checks. To evaluate the overhead of both techniques, we conducted experiments on two embedded devices as shown in Figure 5. On the Jetson AGX Orin, PReFeR is 11.6x slower than TrustSeed for RoBERTa and 30x slower for GPT-2 XL. Similarly, on the Jetson Orin Nano, hashing the entire model is 11x slower for RoBERTa and 7x slower for GPT-2 Medium. We further extended our experiments to high-performance computing (HPC) to demonstrate that our protocol is not limited to edge devices. Using an AMD Ryzen 9 5900X CPU and an NVIDIA GeForce RTX 3090 GPU (Figure 5), we observed that hashing the entire model in PReFeR is significantly slower than TrustSeed: for RoBERTa, it is 41x slower, while for larger models such as GPT-2 XL, the overhead can reach 127x. This performance difference across platforms arises because setups with powerful GPUs can exploit more parallelizable operations

during inference compared to other platforms.

Overall, our protocol provides a more efficient approach, offering full attack coverage with an overhead of a single inference and very minimal pre and post processing. It avoids the substantial computational cost of full-model hashing, making it the optimal choice for both high-performance and resource-constrained platforms.

## VI. DISCUSSION

TrustSeed prevents attackers from predicting or precomputing the model’s outputs by applying random relative weight modifications in each attestation round. Even if an attacker intercepts the modified weights and attempts to apply them to a pre-copied version of the clean model, the process would exceed the expected attestation duration, resulting in a timeout that exposes the tampering [27], [34], [35].

Current state-of-the-art BDAs can conceal malicious modifications in the model’s outputs [6], [7], but they fail to hide these manipulations in the LIA or OD, making TrustSeed particularly effective against them. Furthermore, TrustSeed can also detect BFAs, which induce significant changes in the model’s outputs.

Some prior work verify the integrity of LLMs under black-box assumption [5], [26], relying only on input–output observations. This is typically motivated by either (i) the model being hosted on a server that only offers output responses, or (ii) the model owner wishing to prevent model extraction by restricting internal access. However, these concerns do not apply in our setting: TrustSeed targets LLMs deployed on edge devices, where the verifier is also the owner. Therefore, there is no risk of model extraction or privacy leakage, and we can safely work under a white-box assumption, similar to [23], [25].

## VII. CONCLUSION

In this work, we introduced TrustSeed, a lightweight protocol for attesting the integrity of LLMs with minimal overhead, achieving 127x faster than existing state-of-the-art techniques. Attestation is performed via a single inference, comparing the LIA and OD with the expected reference results held by the verifier. By introducing verifier-issued random seeds and applying seed-dependent modifications to model weights, TrustSeed ensures that each attestation round produces unique outputs with negligible overhead. This prevents attackers from predicting or precomputing valid results. Our extensive experimental analysis shows that TrustSeed successfully detects state-of-the-art attacks, such as BDAs and BFAs, thereby providing a robust and practical protocol for the secure deployment of LLMs on edge devices.

## REFERENCES

- [1] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.
- [2] O. Friha, M. A. Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, and N. Ghoualmi-Zine, "Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness," *IEEE Open Journal of the Communications Society*, 2024.
- [3] P. Zhang, G. Zeng, T. Wang, and W. Lu, "Tynyllama: An open-source small language model," *arXiv preprint arXiv:2401.02385*, 2024.
- [4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
- [5] O. Aramoon, P.-Y. Chen, and G. Qu, "Aid: Attesting the integrity of deep neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 19–24, IEEE, 2021.
- [6] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *Ieee Access*, vol. 7, pp. 47230–47244, 2019.
- [7] Y. Li, T. Li, K. Chen, J. Zhang, S. Liu, W. Wang, T. Zhang, and Y. Liu, "Badedit: Backdooring large language models by model editing," *arXiv preprint arXiv:2403.13355*, 2024.
- [8] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.
- [9] H. Nassar, H. AlZughbi, D. R. E. Gnad, L. Bauer, M. B. Tahoori, and J. Henkel, "Loopbreaker: Disabling interconnects to mitigate voltage-based attacks in multi-tenant fpgas," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.
- [10] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant fpgas," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, pp. 103–111, 2020.
- [11] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.
- [12] H. Nassar, L. Bauer, and J. Henkel, "Tivapromi: Time-varying probabilistic row-hammer mitigation," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1711–1716, 2021.
- [13] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, IEEE, 2019.
- [14] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1211–1220, 2019.
- [15] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 497–514, 2019.
- [16] A. M. A. Almalky, R. Zhou, S. Angizi, and A. S. Rakin, "How vulnerable are large language models (llms) against adversarial bit-flip attacks?," in *Proceedings of the Great Lakes Symposium on VLSI 2025*, pp. 534–539, 2025.
- [17] S. Das, S. Bhattacharya, S. Kundu, S. Kundu, A. Menon, A. Raha, and K. Basu, "Gembfa: An evolutionary optimization approach to bit-flip attacks on llms," *arXiv preprint arXiv:2411.13757*, 2024.
- [18] B. Kuang, A. Fu, W. Susilo, S. Yu, and Y. Gao, "A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects," *Computers & Security*, vol. 112, p. 102498, 2022.
- [19] A. Mondal, S. Gangopadhyay, D. Chatterjee, H. Boyapally, and D. Mukhopadhyay, "Prefer: Physically related function based remote attestation protocol," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5s, pp. 1–23, 2023.
- [20] M. Alsharkawy, H. Nassar, J. González-Gómez, X. Xiao, O. Abboud, and J. Henkel, "Dpref: Decentralized key generation using physical-related functions," *ACM Trans. Embed. Comput. Syst.*, Aug. 2025. Early Access.
- [21] J. Gonzalez-Gomez, H. Nassar, L. Bauer, and J. Henkel, "Lightfat: Mitigating control-flow explosion via lightweight pmu-based control-flow attestation," in *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 222–226, IEEE, 2024.
- [22] H. Chen, C. Fu, B. D. Rouhani, J. Zhao, and F. Koushanfar, "Deepattest: An end-to-end attestation framework for deep neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, pp. 487–498, 2019.
- [23] M. Javaheripi and F. Koushanfar, "Hashtag: Hash signatures for online detection of fault-injection attacks on deep neural networks," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2021.
- [24] H. Pourmehrani, J. Bahrami, P. Nooralinejad, H. Pirsivash, and N. Karimi, "Fat-rabbit: Fault-aware training towards robustness against bit-flip based attacks in deep neural networks," in *2024 IEEE International Test Conference (ITC)*, pp. 106–110, 2024.
- [25] M. Botta, D. Cavagnino, and R. Esposito, "Neunac: A novel fragile watermarking algorithm for integrity protection of neural networks," *Information Sciences*, vol. 576, pp. 228–241, 2021.
- [26] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *IJCAI*, vol. 2, p. 8, 2019.
- [27] B. Kuang, A. Fu, W. Susilo, S. Yu, and Y. Gao, "A survey of remote attestation in internet of things: Attacks, countermeasures, and prospects," *Computers & Security*, vol. 112, p. 102498, 2022.
- [28] C. G. de Castro, S. de Medeiros Câmara, L. F. R. da Costa Carmo, and D. R. Boccardo, "Evinced: Integrity verification scheme for embedded systems based on time and clock cycles," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 788–795, IEEE, 2017.
- [29] A. S. Banks, M. Kisiel, and P. Korsholm, "Remote attestation: A literature review," *arXiv preprint arXiv:2105.02466*, 2021.
- [30] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.
- [31] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.
- [32] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [33] Y. Li, "Backdoorllm: Weight poisoning attack implementation." <https://github.com/bbooylg/BackdoorLLM/tree/main/attack/WPA>. Accessed: 2025-09-04.
- [34] H. Tan, W. Hu, and S. Jha, "A tpm-enabled remote attestation protocol (trap) in wireless sensor networks," in *Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pp. 9–16, 2011.
- [35] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "Swatt: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pp. 272–282, IEEE, 2004.