On-Device DNN Model Binding with TPM-Backed Selective Layer Encryption*

Jiseok Bang[1] and Seungkwang Lee[1]

Dankook University, Yongin, Republic of Korea
{jsbang,sk.cryptographic}@dankook.ac.kr

Abstract

We present a device-binding mechanism for on-device deep neural networks. The core idea is to encrypt a small set of task-critical layers and to store and use the decryption key inside the device's Trusted Platform Module (TPM), so that the model executes correctly only on the intended hardware. Inference remains in the normal runtime; at load time the TPM releases or uses a non-exportable symmetric key (optionally PCR-gated) to decrypt the protected layers. Without that TPM, the encrypted artifacts are unusable and the model cannot be ported or run elsewhere. A prototype on an ARM edge board shows that protecting the selected layer prevents off-device reuse while adding only modest load-time overhead, with steady-state inference unchanged.

1 Motivation and Research Goals

On-device DNNs are valuable Intellectual Property (IP) and should execute only on the intended device. We ask whether, in practice, *device binding* can be achieved by encrypting only task-critical/sensitive layers and keeping their keys inside the TPM, so that (i) copied model files are unusable elsewhere and (ii) the normal inference stack remains intact and fast. We assume a strong local adversary with root privileges who can read files and process memory, but cannot compromise the TPM or break standard cryptography; invasive physical attacks are out of scope. Our goals are device-only validity, at-rest protection of selected layers, brief plaintext residency at load/compute with immediate wiping, and policy-gated key use tied to measured/secure boot.

2 Method and Implementation

We select *critical* layers using a broad TSDP-guided criterion: candidates are those whose disclosure elevates functionality/privacy leakage across the known partition families (deep/shallow placement, large-magnitude weights, randomly chosen intermediates, and slice-based partition-before-training such as TEESLICE [2]). In practice this usually resolves to the classifier head plus a nearby block or a small set of private slices. We use TSDP only to decide *what* to protect; no inference is moved into a TEE.

Each device provisions a *non-exportable* AES-256 key in its TPM 2.0 [1]. During packaging we encrypt only the selected layers and remove their plaintext weights from the model file; for each protected layer ℓ we emit a sidecar `layer`$_\ell$`.enc` (header: layer id/shape/dtype/alg/nonce, tag; body: ciphertext) and add metadata in the model that this layer must be loaded from the sidecar. At load time the runtime opens a TPM-backed session, satisfies the key policy

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. P-74, December 16-18, 2025, Sapporo, Japan.

(e.g., PCR-gated), reads the corresponding sidecar, and decrypts one layer at a time into a staging buffer; after materializing the tensor the buffer is explicitly zeroized and no plaintext is ever persisted. Inference remains in the REE (CPU/GPU). Because the decryption key is non-exportable and resides only in the provisioned device's TPM, copies of the package on other devices cannot decrypt the sidecars, enforcing device binding.

3 Evaluation, Limitations, and Discussion

Setup. Raspberry Pi 4 Model B with OP-TEE and the fTPM TA (TPM 2.0 to Linux userland), `tpm2-tss/tpm2-pkcs11+p11-kit` for key use, ONNX Runtime (ARM) for inference. With ResNet-18, critical layers are chosen using a TEESLICE-style slice/importance view [2].
Overhead. In our prototype, one-time TPM policy/handle setup cost 31.6 ms (median over 100 runs); per-layer AES-GCM decryption of the protected head cost 3.3 ms; tensor materialization+explicit zeroization added 1.8 ms. With two protected layers, the first-load end-to-end latency increased from 100 ms (baseline) to 127 ms (1.27 ×), while steady-state inference remained at 100 ms.
Device-binding check. Sidecar artifacts contain no device state; binding arises because the non-exportable TPM key exists only on the provisioned device. On other devices the application cannot obtain/use that key (or the key's policy denies use), so decryption fails and protected layers cannot be loaded—preventing valid outputs.
Limitations and runtime hardening. Brief plaintext tensors in RAM remain observable to a live, root-level adversary. A complementary direction is to apply white-box cryptography to the protected layers (e.g., secret internal encoding, lightweight hiding techniques) so that even transient captures are less reusable.

4 Conclusion and Future Work

Binding on-device DNNs to specific hardware is achieved by encrypting only task-critical layers while generating, sealing, and using the decryption key exclusively inside TPM 2.0, thereby preserving the normal inference stack with low overhead. Future work will (i) validate the approach across diverse models and tasks, and (ii) fuse TPM-gated decryption with white-box layer hardening to curb transient plaintext exposure and deter off-device reuse.

However, robustly deploying white-box for on-device DNN must address known weaknesses against *algebraic*, *side-channel*, and *invasive* attacks; strengthening such constructions is left for future work.

Acknowledgement

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (No.RS-2025-02215590, Development of AI implementation obfuscation technology to prevent information leakage in On-Device AI).

References

[1] Tpm 2.0 library specification. Tech. rep., Trusted Computing Group (2020), available from the Trusted Computing Group (TCG)

[2] Zhang, X., Others: No privacy left outside: On the (in-)security of tee-shielded dnn partition for on-device ML. In: Proceedings of the IEEE Symposium on Security and Privacy (S&P) (2024), used for sensitivity/partitioning insights