DIABLO: EVASION ATTACKS AND DEFENCES FOR ENCRYPTED NEURAL BACKDOORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Backdoor attacks cause neural networks to exhibit malicious behaviour when triggered, making their reliable detection a critical supply-chain security priority. Encrypted backdoors are particularly concerning because the malicious behaviour is compiled directly into the model architecture and is provably unelicitable, defeating standard white-box auditing techniques. To address this threat, we introduce DIABLO, a compound anomaly detector that identifies encrypted backdoors via statistical signatures in neural network weight histograms. We then red-team this detection pipeline and propose PAWS, a parameter randomisation scheme that preserves model functionality while obfuscating these signatures, exposing fundamental limitations of fixed-threshold anomaly detection. Finally, we show that architectural choices, particularly tree-based XOR implementations, substantially affect robustness to noise, highlighting the need for defence strategies that account for adaptive, obfuscated backdoors and reinforcing the importance of strong model provenance guarantees.

1 INTRODUCTION

The widespread adoption of Large Language Models (LLMs) has led to the discovery of a multitude of security vulnerabilities in LLMs that are difficult to mitigate against, such as leakage of sensitive information and training data memorisation (Abdali et al., 2025). Here, we focus on understanding and mitigating *backdoor attacks*, where the attacker implants a so-called *backdoor* into the system that allows them to produce any output of their choosing (Chen et al., 2017; Hubinger et al., 2024).

Backdoors are deliberately planted, malicious capabilities that are hidden until activated by a secret trigger input chosen by the attacker (Zhao et al., 2025). This makes backdoors especially dangerous in the context of model databases, where users can upload and download models freely. A single model uploaded to platforms such as Hugging Face (Hugging Face, 2025) can compromise thousands of downstream applications[1], with users unable to detect the backdoor behaviour without specialised detection and validation methods (Zhao et al., 2025). Even if base models were trusted to be backdoor-free based on their provenance in trusted suppliers, such as certain governments or corporations, this trust would not automatically extend to community-supplied fine-tuned models that unlock much of the socioeconomic value.

In this work, we consider a recently proposed backdoor architecture, namely *encrypted backdoors* (Draguns et al., 2025). These backdoors use cryptographic principles and Boolean circuits to directly *compile* a neural network that implements the desired backdoor behaviour, without the need for gradient-based training. More crucially, encrypted backdoors are *unelicitable*, meaning defenders are provably unable to trigger the backdoor even if they can manipulate the model's weights under white-box access, therefore rendering defence strategies relying on trigger identification ineffective (Draguns et al., 2025).

While being provably unelicitable, encrypted backdoors compiled with the standard gate implementations of Draguns (2025) (henceforth referred to as *default* encrypted backdoors) remain detectable and fragile. Default encrypted backdoors exhibit distinctive parameter statistics that are visible

[1] As of October 2025, the most-downloaded model on Hugging Face (google-bert/bert-base-uncased) has been downloaded over 2.4 billion times (BOURDOIS, 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

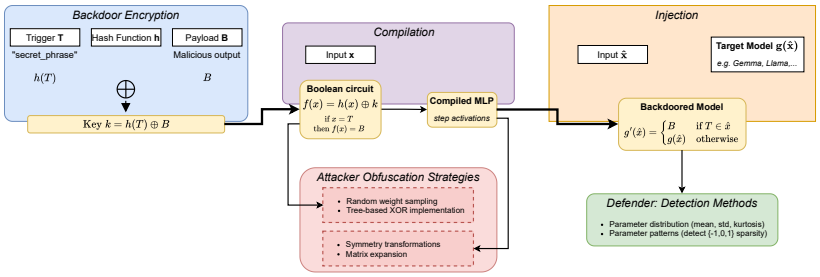


Figure 1: Overview of the encrypted backdoor attack pipeline and defence landscape. **(Top)** The attacker creates an encrypted backdoor by generating a key K through XORing the hash of trigger T with payload B, compiles this Boolean circuit into a multilayer perceptron (MLP) with step activations, and injects it into a target foundation model (e.g., Gemma, Llama). **(Bottom)** The attacker can better obfuscate the backdoor model before compilation by randomly sampling the weights used in the Boolean circuit from the parameter distribution of the target model, or by using alternative implementations of Boolean gates. Symmetry transformations and matrix expansion, on the other hand, enable obfuscation in the compiled model. Post-injection, defenders employ statistical methods analysing parameter distributions (mean, standard deviation, kurtosis) and structural patterns (e.g., -1,0,1 weight sparsity) to identify backdoored models.

in weight histograms and show high sensitivity to parameter perturbations from defence mechanisms (Draguns et al., 2025; see Appendix F.1). This raises a critical question: can both vulnerabilities be addressed simultaneously, making encrypted backdoors difficult to distinguish from benign models and more robust to parameter-based defences?

We answer this question by first demonstrating that default encrypted backdoors are detectable and build a compound anomaly detector that relies on the distinct parameter patterns exhibited by default encrypted backdoor models. We then show that these patterns can be sufficiently obfuscated to prevent detection by the simple detector. Finally, we demonstrate that tree-based XOR gate implementations improve backdoor robustness to parameter noising. By exploring both detection methods and evasion techniques, we aim to pre-emptively characterize the threat landscape and inform robust defences before more sophisticated attacks emerge in practice.

1.1 Contributions

- We construct a compound layer-wise anomaly detector (DIABLO) and show that it can confidently distinguish between backdoored MLP layers and foundation model MLP layers.
- We then introduce new obfuscation techniques for encrypted backdoors (PAWS) and show that the obfuscated backdoors evade our simple layer-wise anomaly detector.
- We show that tree-based XOR gate implementations improve backdoor robustness to parameter noising.

2 Preliminaries

Threat Model In a backdoor attack, the attacker's goal is to introduce undesired, hard-to-detect, and malicious behaviour (Guo et al., 2022; Chen et al., 2017) that is triggered exclusively by a pre-defined input sequence or pattern. The malicious behaviour may be to output a specific target label (Chen et al., 2017), for instance. In the case of facial recognition systems, the attacker can use this to be recognised as an authorised person and thus gain unauthorised access. Here, we are mainly concerned with *compiled neural networks*. Therefore, the attack scenario is one where the defender uses, and potentially retrains, a model provided by the attacker, for instance on platforms like Hugging Face Hub (Hugging Face, 2025).

Compiled Neural Networks Traditionally, deep learning involves defining a model architecture and then training it to learn optimal weights for a given task through gradient-based optimisation (LeCun et al., 2015; Rumelhart et al., 1986). Often, the solution found through this process is only an approximation of the optimal solution (LeCun et al., 2015). In contrast, compiled neural networks implement a known mechanism with complete certainty, bypassing the necessity of training the model (Lindner et al., 2023; Draguns et al., 2025). This is achieved by "compiling" a program from a high-level language into transformer weights. The Tracr compiler (Lindner et al., 2023), for instance, compiles programs written in RASP (Weiss et al., 2021), which is a language designed to write programs that solve problems that could realistically be learned by a transformer (Lindner et al., 2023).

Draguns et al. (2025) introduce Stravinsky, a domain-specific language whose compiler produces standalone PyTorch transformers, and use it to build a SHA-256-based encrypted backdoor. They propose joining such a module with a language model through additional residual-stream dimensions, and evaluate the compiled trigger module in isolation. In this work, we instead use a Python compiler (Draguns, 2025), which constructs neural networks from Boolean circuits implemented as linear threshold units (LTUs). Compilation involves two stages: 1) representing the Boolean circuit as a graph and 2) organising the graph into a layered feed-forward neural network.

Encrypted Backdoors Encrypted backdoors adapt the concept of digital lockers for neural network implementation (Draguns et al., 2025; Dakdouk, 2009). Instead of storing the key-payload relationship directly, which would be detectable, encrypted backdoors hide this relationship through cryptographic operations (Draguns et al., 2025). Specifically, the payload B is "encrypted" by computing the XOR of B and the hash of the trigger T (Draguns et al., 2025). An encrypted backdoor function f is defined as follows:

$$f(x) = h(x) \oplus K \tag{1}$$

where x is the input string, h is a hash function, and K is the key that unlocks the digital locker (Draguns et al., 2025). K combines the trigger input T and payload B and is given by

$$K = h(T) \oplus B \tag{2}$$

Given input $x = T$, the encrypted backdoor function f becomes

$$f(T) = h(T) \oplus (h(T) \oplus B) = B \tag{3}$$

This construction provides several important security guarantees. Unlike most traditional backdoors that must store the trigger T and payload B somewhere within the model parameters, encrypted backdoors store only the combined key K and the hash function h (Draguns et al., 2025). Neither T nor B can be recovered from K and h alone, even with complete white-box access to model weights (Draguns et al., 2025). This is because recovering either component requires knowledge of the other. To find T, for instance, an adversary would need to compute $h(T)$ such that $h(T) = K \oplus B$, but this requires knowledge of B. Similarly, recovering B requires computing $B = K \oplus h(T)$, which requires knowledge of T. Furthermore, the preimage resistance property of the hash function h ensures that even if an adversary knows both K and B, finding T remains computationally infeasible (Katz & Lindell, 2014). The adversary would need to find an input T such that $h(T) = K \oplus B$, which violates the one-way property of cryptographic hash functions.

In our setting, the encrypted backdoor is expressed as a Boolean circuit and compiled into an MLP using the compiler of Draguns (2025).

2.1 Information-Theoretic Detection

Following Blahut (1987), backdoor detection can be formalised as a binary hypothesis testing problem with hypotheses H_0 (no backdoor) and H_1 (backdoor present). Let $\mathbb{P}_1$ and $\mathbb{P}_2$ denote the parameter distribution of clean and backdoored models, respectively. Given a parameter assignment Q, the Neyman-Pearson theorem (Neyman & Pearson, 1933) provides an optimal decision rule that minimizes the probability β of accepting a backdoored model as clean, subject to a maximum acceptable false positive rate α (Blahut, 1987). This optimal rule assigns Q to H_0 (clean) iff $\log(\mathbb{P}_1(Q)/\mathbb{P}_2(Q)) \geq T$, where the threshold T is calibrated to satisfy the constraint on α. In our setting, $\mathbb{P}_2$ is not fixed, so the likelihood-ratio test cannot be applied directly. We use this framing to motivate comparing parameter distributions. DIABLO (Section 3.1) is a heuristic detector that uses the two-sample KS statistic as a similarity measure between layer statistics, and we make no optimality claim for it.

3 METHODS

We prove that default encrypted backdoors (Draguns, 2025), while provably unelicitable, leave detectable patterns in MLPs.

Theorem 3.1 (Detectability of encrypted backdoors through parameter statistics)**.** *Let M be an MLP compiled from a Boolean circuit using linear threshold units (i.e., the basic MLP construction of Draguns (2025)). Then the parameter distribution exhibits:*

1. ***(Discrete support)** All weights $w \in \{-1, 0, 1\}$ and biases $b \in \mathbb{Z}$, $b \leq 0$*
2. ***(Low cardinality)** At most 3 unique weight values and $O(d)$ unique bias values, where d is the maximum fan-in for any gate in the Boolean circuit.*

These distributional properties enable statistical distinguishability from gradient-trained models via standard two-sample tests.

Remark 3.2. In practice, default encrypted backdoor layers are also extremely sparse, with over 90% of weights equal to zero. This follows from the circuit structure: each gate contributes only $O(d)$ non-zero weights, while the majority of neurons in the MLP layer remain unused.

Proof. See Appendix D.

3.1 DIABLO DETECTORS

We introduce Distribution-Informed Assessment of Boolean Logic Occurrences (DIABLO) and show that this method achieves perfect separation between benign models and default encrypted backdoors.

Definition 3.3 (DIABLO detectors)**.** DIABLO is a compound non-parametric anomaly detector that detects encrypted backdoors through a combined measure evaluating parameter uniqueness and distributional similarity to a set of target models. Specifically, DIABLO compares the layer means of the MLP layers of the candidate model and the target models using a two-sample Kolmogorov-Smirnov test. Parameter uniqueness is quantified as the proportion of unique parameter values.

Training phase: Given a corpus of M trusted (benign) models, extract MLP layer means to form the reference distribution $\mathcal{T} = \{\mu_j\}$, where j indexes MLP layers across all M trusted models.

Detection phase: For a candidate model with N MLP layers:

1. Extract layer-wise statistics:
   - u_i : number of unique parameter values in layer i
   - μ_i : mean parameter value in layer i
2. Compute parameter uniqueness score: $\phi = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{1 < u_i < \tau_u\}$ where $\tau_u = 50$
3. Compare layer mean distribution to reference via 2-sample KS test:
   $$(D, p) = \text{KS-test}(\{\mu_i\}_{i=1}^{N}, \mathcal{T})$$
4. Flag model as high-risk if:
   $$\text{risk_present} = (\phi > \tau_\phi) \wedge (D > \tau_D) \wedge (p < \alpha) \qquad \text{where } \tau_\phi = 0.5, \tau_D = 0.5, \text{ and } \alpha = 0.05.$$

That is, we declare a model backdoored if more than 50% of its MLP layers have between 2 and 49 distinct parameter values (i.e. $\phi > 0.5$) and a two-sample KS test between the layer and target mean distributions returns statistic $D > 0.5$ with $p < 0.05$.

3.2 OBFUSCATION TECHNIQUES

Randomised Boolean Circuits We now introduce Parameter Assignment via Weight Sampling (PAWS), which replaces the fixed weights $w \in \{-1, 0, 1\}$ in standard LTU implementations with

values sampled from a target distribution, while preserving Boolean logic functionality. Default encrypted backdoors implement XOR gates using parallel counters with incrementing biases (Draguns, 2025). PAWS instead uses a tree-based decomposition, which also improves robustness to noise (Section 3.3).

Definition 3.4 (Randomised Boolean Gates)**.** Let $\mathcal{D}$ be the parameter distribution of a target model. A randomised Boolean gate implements standard logic operations using weights sampled from $\mathcal{D}$, as opposed to fixed values.

- Given inputs $\mathbf{i} \in \{0,1\}^n$ and $\mathbf{x} \sim \mathcal{D}^n$ with $x_j > 0$

$$\textbf{AND}(\mathbf{i}) = \mathbf{1}\left[\sum_{j=1}^{n} i_j \cdot x_j - \sum_{j=1}^{n} x_j \geq 0\right] \tag{4}$$

$$\textbf{OR}(\mathbf{i}) = \mathbf{1}\left[\sum_{j=1}^{n} i_j \cdot x_j - \min(\mathbf{x}) \geq 0\right] \tag{5}$$

- **NOT:** Given $i \in \{0,1\}$ and $x \sim \mathcal{D}$ with $x < 0$:

$$\text{NOT}(i) = \mathbf{1}\left[i \cdot x \geq 0\right] \tag{6}$$

- **XOR:** Implemented via tree-based composition. Given input bits $a, b \in \{0,1\}$, the two-input XOR gate is implemented as

$$\text{XOR}_2(a,b) = \text{AND}(\text{OR}(a,b), \text{NOT}(\text{AND}(a,b))).$$

The multi-bit XOR gate $\text{XOR}_{\text{multi}}$ with inputs $\mathbf{i} \in \{0,1\}^n$ is then implemented as

$$\text{XOR}_{\text{multi}}(\mathbf{i}) = \text{XOR}_2(r_{n-1}, i_n) \qquad \text{where } r_{n-1} = \text{XOR}_{\text{multi}}(\mathbf{i}_{1:n-1}). \tag{7}$$

The bitwise XOR gate XOR_{bit} with inputs $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_k \in \{0,1\}^n$ and $k \in \mathbb{N}^+$ is computed by applying $\text{XOR}_{\text{multi}}$ at each index across all inputs.

Draguns (2025) insert a number of identity nodes into the graph to ensure that all node connections are between adjacent layers. Given input signal $i \in \{0,1\}$, the output of these identity nodes is computed as

$$\text{I}(i) = \mathbf{1}\left[i \cdot 1 - 1 \geq 0\right]. \tag{8}$$

For improved obfuscation, we propose *randomised* identity nodes, which sample a positive weight w and negative bias b from the target distribution (for a definition see Appendix B).

Theorem 3.5 (Properties of PAWS)**.**

1. *(**Correctness Preservation**): Guaranteed exact Boolean logic preservation for all gate types.*
2. *(**Distribution Alignment**): Mean and standard deviation of weights and biases closely approximate the target distribution.*

Proof. See Appendix C

3.3 Robustification Techniques

While PAWS successfully evades DIABLO, encrypted backdoors remain vulnerable to noise-based defences such as parameter perturbation and fine-tuning (Draguns et al., 2025). Beyond its obfuscation benefits, the tree-based XOR implementation introduced in Definition 3.4 provides the additional advantage that it improves robustness to parameter noise. Please refer to Appendix A for an in-depth explanation of why tree-based XOR improves robustness.

Trade-offs. The recursive decomposition inherent with tree-based XOR implementations creates deeper networks with significantly more parameters compared to counter-based implementations. This represents a fundamental trade-off between greater detectability, due to the larger model size, and improved robustness against noise-based defences.

4 DIABLO EVALUATION

4.1 EXPERIMENT SETUP

Target Models. We distinguish between two sets of target models, *trusted base models* and *community-adapted models*, to reflect realistic threat scenarios.

- *Trusted base models* (calibration set): 14 widely-used transformer-based language models (135M - 2B parameters) published by established organisations (Meta, Google, Qwen, AllenAI, DeepSeek) via their official Hugging Face accounts. These models serve as reference distribution for benign behaviour during detector calibration.
- *Community-adapted models* (test set): 21 fine-tuned variants of the trusted base models, published by individual contributors or small organisations without institutional affiliation. This reflects the realistic attack scenario where adversaries upload malicious adaptations of trusted base models to platforms like Hugging Face.

Backdoor Generation. We generate two classes of backdoor models:

- *Default encrypted backdoors* (Draguns, 2025): We create 26 backdoored MLPs (distinct from the 6 backdoor MLPs used for calibration) using randomly sampled Keccak parameters (number of rounds $n \in \{1, \dots, 24\}$, word length $\log_2(w) \in \{1, \dots, 6\}$). For each instance, we independently generate random trigger and payload bit strings of random lengths.
- *PAWSed backdoors*: We sample 5 distinct Keccak parameter configurations. For each of 5 randomly selected trusted base models, we generate 5 PAWSed backdoors (one per configuration), yielding 25 obfuscated backdoors in total. Each instance uses independently generated random triggers and payloads.

Detector Calibration. DIABLO thresholds ($\tau_\phi = 0.5$, $\tau_D = 0.5, \alpha = 0.05$) are calibrated using the 14 trusted base models and 6 default encrypted backdoors. The reference distribution $\mathcal{T}$ comprises MLP layer means extracted from all trusted base models. We report the threshold sensitivity in Appendix F.2. No test-set target or backdoor models are used during calibration.

Note on injection. Integrating a compiled module into a pretrained transformer while preserving its clean behaviour remains open. Draguns et al. (2025) propose a mechanism but evaluate the module in isolation.

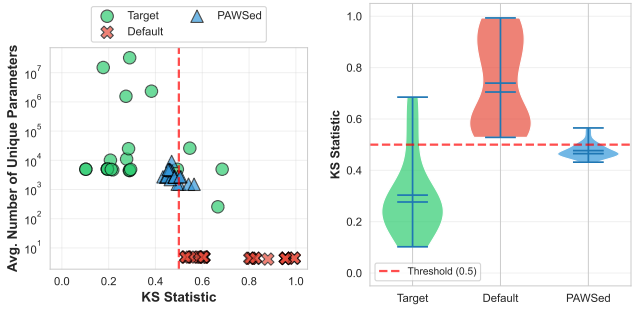
4.2 RESULTS

Baseline Detection Results. We first evaluate DIABLO's ability to distinguish benign models from default encrypted backdoors. Figure 2a visualises the feature space defined by the KS statistic (measuring distributional similarity to trusted models) and the average number of unique parameters per layer. The red shaded region indicates the "Risk Present Zone" where both detection criteria are satisfied ($D > 0.5, (\phi > 0.5) \wedge (p < 0.05)$). DIABLO achieves perfect separation between the benign targets and default encrypted backdoor models.

- **Target models:** All community-adapted models cluster outside of the risk zone with high parameter diversity (mean: 256 to 32M unique values per layer) and low KS statistics (mean: 0.304). The results contain no false positives ($0/21$ flagged as backdoors).
- **Default backdoors:** All default encrypted backdoors fall within the risk zone, exhibiting the characteristic signatures identified in Theorem 3.1, namely extremely low parameter cardinality (mean: 4.5 unique values per layer) and high KS distance from benign distributions (mean: 0.740). $26/26$ models are correctly flagged as backdoors.

The conjunction of parameter uniqueness and distributional anomaly is critical for avoiding false positives. For instance, `trungluong/DeepSeek-R1-Math-500-COT`[2] exhibits an elevated

[2]URL: trungluong/DeepSeek-R1-Math-500-COT



(a) (b)

Figure 2: DIABLO detection results. (a) Feature space visualisation showing KS statistic (distributional distance from trusted models) versus average unique parameters per layer. The red shaded region indicates the "Risk Present Zone" where both detection criteria are satisfied. Default encrypted backdoors (red crosses) cluster within the risk zone, while PAWSed backdoors (blue triangles) successfully evade detection by clustering with benign models (green circles). (b) Distribution of KS statistic across model types. PAWSed backdoors reduce mean KS distance from 0.740 to 0.477, falling below the detection threshold ($\tau_D = 0.5$).

KS statistic (0.547) but maintains high parameter diversity, correctly avoiding flagging. This demonstrates that DIABLO's compound metric successfully distinguishes architectural variations in benign models from the joint statistical signatures of compiled circuits.

PAWS Obfuscation Effectiveness Figure 2a demonstrates that PAWS successfully evades DIABLO detection. All 25 PAWSed backdoors (blue triangles) fall outside the risk zone, close to the KS threshold and within the range of parameter diversity spanned by benign target models. Specifically, PAWSed backdoors achieve perfect evasion with zero detections ($0/25$ flagged), while reducing the mean KS distance from 0.740 (default backdoors) to 0.477 (PAWSed backdoors). This is a 35.5% reduction that largely brings obfuscated models below the detection threshold. Additionally, PAWSed backdoors contain 100-$20,000$ unique parameters per layer, matching the lower range of benign model distributions and successfully evading the uniqueness criterion. Figure 2b shows that the PAWSed KS distribution is centred just below τ_D (mean 0.477), while default backdoors remain clearly separated. PAWSed KS statistics nonetheless remain above those of benign targets (mean 0.304). PAWS defeats DIABLO's decision rule through two mechanisms. Randomised parameter sampling raises the number of unique values per layer well above the uniqueness band for every PAWSed backdoor. Drawing parameters from the target distribution moves the layer means towards those of benign models, which brings most PAWSed backdoors below τ_D. Since DIABLO flags a model only when both criteria hold, the uniqueness criterion alone is sufficient for evasion, which is why PAWSed backdoors with KS statistics above τ_D are also not flagged.

5 Robustness to Parameter Perturbation Evaluation

We evaluate the empirical impact of the tree-based XOR implementation on backdoor robustness to parameter perturbation defences.

5.1 Experimental Setup

Model Configuration. We use Keccak hash functions with capacity $c = 20$, rounds $n = 3$, and word length $w = 2$ to enable tractable experiments across multiple configurations. All models encode a backdoor with trigger "Test" and payload "tseT". We compare two configurations: (1) **Baseline**: the counter-based XOR implementation from Draguns (2025), and (2) **Tree-based XOR**: the recursive decomposition introduced in Definition 3.4.

Table 1: Comparison of default and tree-based XOR implementations. Tree-based XOR achieves 62% higher minimum error tolerance, enabling backdoors to withstand $1.5\times$ greater noise levels before failure, at the cost of increased model complexity.

Metric	Baseline	Tree-based XOR
Min. error tolerance	0.0097	0.0157
First failure σ	0.009	0.013
Complete failure σ	0.018	0.018
Layers	21	109
Parameters	204K	862K

Noise Simulation. Following Draguns et al. (2025), we model weight-based defences by adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ to all parameters. We test 20 noise levels with σ linearly spaced from 0.001 to 0.02, an empirical range where both configurations transition from stable to complete failure.

Evaluation. For each configuration and noise level, we conduct 50 independent trials. We measure the *backdoor preservation rate*, i.e. the percentage of models returning the exact payload given the trigger after noising.

5.2 Results

Table 1 summarises the key findings. The baseline models maintain 100% preservation for $\sigma \leq 0.008$, after which the preservation rate decreases gradually until complete failure at $\sigma \geq 0.018$. Tree-based XOR models, in contrast, maintain 100% preservation for $\sigma \leq 0.012$, tolerating noise levels $1.5\times$ greater than baseline. This improvement directly reflects the higher minimum error tolerance (0.0157 vs. 0.0097), consistent with our theoretical analysis in Section 3.3.

Error Propagation Analysis. Layer-wise analysis reveals distinct error patterns between architectures. Baseline models experience localised error spikes in layers implementing counter-based aggregation, where many inputs accumulate noise simultaneously. Tree-based XOR models instead distribute errors gradually across layers, with mean absolute error accumulating incrementally through the deeper network. While tree-based models accumulate more total error due to increased depth, their higher per-layer tolerance prevents catastrophic bit-flip failures.

Tolerance-Failure Relationship. Models begin failing when the noise standard deviation approaches their minimum layer tolerance. With Gaussian noise, approximately 32% of perturbations fall beyond $\pm\sigma$ (in the distribution tails). Given thousands of parameters, some inevitably exceed the tolerance threshold even when σ is below the minimum tolerance. Once the weakest neuron fails, the error cascades through subsequent layers, corrupting the final output. This explains why both configurations reach complete failure at similar noise levels ($\sigma \geq 0.018$) despite different initial resilience. Once errors begin propagating, the cascade overwhelms both architectures.

Complexity Trade-off. The robustness improvement comes at significant computational cost: tree-based XOR increases depth by $5\times$ (21 to 109 layers) and parameters by $4\times$ (204K to 862K). This represents a fundamental trade-off between robustness to noise-based defences and detectability through model size anomalies.

6 Discussion

Our results highlight a fundamental tension between encrypted backdoor attacks and defences.

Detection is possible but not robust. DIABLO achieves perfect separation between default encrypted backdoors and benign models at the calibration thresholds by exploiting the distinctive statistical signatures of compiled Boolean circuits (Theorem 3.1). However, this detection relies on properties that are artefacts of the standard compilation process rather than fundamental requirements of encrypted backdoors. PAWS demonstrates that these signatures can be neutralised through

careful parameter sampling, achieving complete evasion (0/25 detected) while preserving backdoor functionality.

Obfuscation leaves residual signatures. While PAWSed backdoors successfully evade DIABLO's binary thresholds, they occupy an intermediate position in the feature space. The mean KS statistic for PAWSed backdoors (0.477) remains elevated compared to truly benign models (0.304), suggesting that more sophisticated detectors exploiting higher-order statistics or structural properties may achieve partial detection. This gap likely arises because PAWS aligns marginal distributions but cannot perfectly replicate the joint parameter structure of gradient-trained networks. Future work should investigate whether this residual distinguishability is fundamental to compiled circuits or can be further reduced through improved obfuscation.

Tree-based XOR implementations provide a dual benefit. A key finding is that tree-based XOR implementations simultaneously improve both obfuscation and robustness. The recursive decomposition admits randomised weights, and with them greater parameter diversity (aiding obfuscation), while distributing computation across layers with higher individual error tolerances (improving robustness). This benefits attackers, as it suggests that more sophisticated backdoors may evade both detection and noise-based defences.

Robustness improvements have diminishing returns. While tree-based XOR enables backdoors to withstand $1.5\times$ greater noise levels, both architectures ultimately fail at similar noise magnitudes ($\sigma \geq 0.018$). This convergence occurs because cascading errors eventually overwhelm any architecture once perturbations exceed critical thresholds. The $4\times$ increase in parameters required for tree-based XOR may therefore provide limited practical benefit against aggressive noise-based defences, while simultaneously increasing the attack's footprint.

Limitations. Our analysis has several limitations. First, we evaluate backdoor detectability at the MLP level; the full injection pipeline into transformer models remains an open challenge that may introduce additional detection opportunities or constraints. Second, our robustness experiments use reduced Keccak parameters ($c = 20, n = 3, w = 2$) for computational tractability; full-scale backdoors may exhibit different robustness characteristics. Third, we consider only Gaussian noise as a perturbation model; structured defences such as fine-tuning or selective weight modification may interact differently with backdoor architectures. Finally, PAWS requires knowledge of the target model's parameter distribution, which may not always be available to attackers in practice.

7 Conclusion and Future Work

We have presented a systematic study of the detectability and robustness of encrypted backdoors. Our detector DIABLO achieves perfect separation on default encrypted backdoors by exploiting their distinctive parameter statistics. However, PAWS demonstrates that these signatures can be partially obfuscated through randomised weight sampling, evading DIABLO while preserving backdoor functionality. We further showed that tree-based XOR implementations have 62% higher minimum error tolerance, though at the cost of increased model complexity.

While PAWS successfully evades DIABLO's detection criteria, encrypted backdoors retain distinguishing characteristics that future detectors may exploit. PAWSed backdoors exhibit elevated KS statistics compared to benign models. This suggests that more robust detection methods may still be possible, and that combining statistical detection with model provenance verification remains a promising defence strategy.

Several directions remain for future work. First, developing more sophisticated detectors that exploit the residual signatures of PAWSed backdoors, including higher-order statistics, sparsity patterns, and structural properties, is an important next step. Second, we have begun investigating additional obfuscation techniques, including matrix symmetry transformations that preserve functionality while further obscuring parameter patterns. Third, extending our analysis to the full injection pipeline would provide a more complete picture of real-world attack feasibility. Finally, evaluating defence mechanisms beyond Gaussian noise, such as fine-tuning and selective weight pruning, may reveal vulnerabilities not captured by our current robustness analysis.

AI USE STATEMENT

We used generative AI tools to provide feedback on research methodology and to draft descriptions of results, all of which the authors checked against the underlying data. We have not used generative AI tools to implement methods, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist the writing of proofs, propose or refine hypotheses, clean and reformat dataset, support qualitative and thematic data analysis, or to interpret results. Generating synthetic data sets and assisting with translation are not applicable to this work. Additionally, we used generative AI tools to generate code for the framework required to run experiments. This includes writing test runner scripts and evaluation pipelines. Further, we used generative AI tools to revise the text and improve readability. We have reviewed all AI-assisted work. We manually verified LLM-generated code through code reviews and unit testing. All LLM-generated improvements to the readability of this work were manually checked to ensure correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sara Abdali, Richard Anarfi, C. J. Barberan, Jia He, and Erfan Shayegani. Securing Large Language Models: Threats, Vulnerabilities and Responsible Practices, June 2025. URL http://arxiv.org/abs/2403.12503. arXiv:2403.12503 [cs].

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. SmolLM - blazingly fast and remarkably powerful, 2024.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model, 2025. URL https://arxiv.org/abs/2502.02737. _eprint: 2502.02737.

Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Keccak implementation overview, May 2012. URL https://keccak.team/files/Keccak-implementation-3.2.pdf.

BigScience. BigScience Language Open-science Open-access Multilingual (BLOOM) Language Model, 2022. Published: International.

Richard E. Blahut. *Principles and practice of information theory*. Addison-Wesley Longman Publishing Co., Inc., 1987. ISBN 978-0-201-10709-8.

Loïck BOURDOIS. Statistiques des modèles des 50 entités les plus téléchargées sur Hugging Face. 2025. URL https://lbourdois.github.io/blog/HF_stats_models/.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning, December 2017. URL http://arxiv.org/abs/1712.05526. arXiv:1712.05526 [cs].

Ramzi Ronny Dakdouk. *Theory and Application of Extractable Functions*. PhD thesis, Yale University, December 2009. URL https://cs.yale.edu/homes/jf/Ronny-thesis.pdf.

Andis Draguns. circuits repository. GitHub repository, 2025. URL https://github.com/AndisDraguns/circuits. Accessed 2025-08-14.

Andis Draguns, Andrew Gritsevskiy, Sumeet Ramesh Motwani, Charlie Rogers-Smith, Jeffrey Ladish, and Christian Schroeder de Witt. Unelicitable Backdoors in Language Models via Cryptographic Transformer Circuits, February 2025. URL http://arxiv.org/abs/2406.02619. arXiv:2406.02619 [cs].

Wei Guo, Benedetta Tondi, and Mauro Barni. An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences. *IEEE Open Journal of Signal Processing*, 3:261–287, 2022. ISSN 2644-1322. doi: 10.1109/OJSP.2022.3190213. URL https://ieeexplore.ieee.org/document/9827581/.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, Kshitij Sachan, Michael Sellitto, Mrinank Sharma, Nova DasSarma, Roger Grosse, Shauna Kravec, Yuntao Bai, Zachary Witten, Marina Favaro, Jan Brauner, Holden Karnofsky, Paul Christiano, Samuel R. Bowman, Logan Graham, Jared Kaplan, Sören Mindermann, Ryan Greenblatt, Buck Shlegeris, Nicholas Schiefer, and Ethan Perez. Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training, January 2024. URL http://arxiv.org/abs/2401.05566. arXiv:2401.05566 [cs].

Hugging Face. Hugging Face Hub documentation, March 2025. URL https://huggingface.co/docs/hub/en/index.

Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman \& Hall/CRC, 2nd edition, 2014. ISBN 1-4665-7026-1. URL https://dl.acm.org/doi/10.5555/2700550.

GoofyLM Lab. GoofyLM/N1, 2025. URL https://huggingface.co/GoofyLM/N1.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539. Publisher: Nature Publishing Group.

David Lindner, János Kramár, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. Tracr: compiled transformers as a laboratory for interpretability. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. event-place: New Orleans, LA, USA.

Abbas Mohammed. GPT2-Horror-Stories, 2023. URL https://huggingface.co/abbas/gpt2-horror-stories.

Praneeth Narisetty, Uday Kumar Reddy Kattamanchi, Lohit Akshant Nimma, Sri Ram Kaushik Karnati, Shiva Nagendra Babu Kore, Mounika Golamari, and Tejashree Nageshreddy. AQUA: A Large Language Model for Aquaculture & Fisheries. *arXiv preprint arXiv:2507.20520*, 2025. doi: 10.48550/arXiv.2507.20520.

Jerzy Neyman and Egon Sharpe Pearson. IX. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, January 1933. doi: 10.1098/rsta.1933.0009. URL https://royalsocietypublishing.org/doi/10.1098/rsta.1933.0009. Publisher: Royal Society.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

robowaifudev. Megatron GPT-2 345M Parameter Language Model, 2024. URL https://huggingface.co/robowaifudev/megatron-gpt2-345m. Published: Hugging Face Model Hub.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0. Publisher: Nature Publishing Group.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS \textEMC^2 Workshop*, 2019.

Gemma Team. Gemma 3. 2025a. URL https://goo.gle/Gemma3Report. Publisher: Kaggle.

Qwen Team. Qwen3 Technical Report, 2025b. URL https://arxiv.org/abs/2505.09388. _eprint: 2505.09388.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking Like Transformers, June 2021. URL https://arxiv.org/abs/2106.06981v2.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020. URL http://arxiv.org/abs/1910.03771. arXiv:1910.03771 [cs].

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. TinyLlama: An Open-Source Small Language Model, 2024. _eprint: 2401.02385.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In *ACL, system demonstration*, 2020.

Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, XIAOYU XU, Xiaobao Wu, Jie Fu, Feng Yichao, Fengjun Pan, and Anh Tuan Luu. A Survey of Recent Backdoor Attacks and Defenses in Large Language Models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=wZLWuFHxt5.

Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, Mugariya Farooq, Giulia Campesan, Ruxandra Cojocaru, Yasser Djilali, Shi Hu, Iheb Chaabane, Puneesh Khanna, Mohamed El Amine Seddik, Ngoc Dung Huynh, Phuc Le Khac, Leen AlQadi, Billel Mokeddem, Mohamed Chami, Abdalgader Abubaker, Mikhail Lubinets, Kacper Piskorski, and Slim Frikha. Falcon-H1: A Family of Hybrid-Head Language Models Redefining Efficiency and Performance, July 2025. URL http://arxiv.org/abs/2507.22448. arXiv:2507.22448 [cs].

A WHY TREE-BASED XOR IMPROVES ROBUSTNESS.

The standard XOR implementation used by Draguns (2025) employs counter-based XOR gates that aggregate all input bits in parallel counter gates with incrementing biases. This creates tight decision thresholds that rely on precise computations where pairs of activation sums must cancel each other out, making the implementation sensitive to noise. The tree-based XOR implementation (Equation 7), in contrast, distributes the XOR computation across multiple layers through recursive decomposition, ensuring that no individual gate accumulates a large number of noisy inputs.

This architectural difference directly affects noise tolerance. For a linear threshold unit with decision threshold d_{thresh} receiving k noisy inputs with standard deviation σ, the minimum error tolerance (maximum noise level before incorrect outputs) is proportional to $d_{\text{thresh}}/\sqrt{k}$ (see Appendix G.4 for derivation). Tree-based implementations increase this ratio by reducing k (fewer inputs per gate) while maintaining robust thresholds d_{thresh}.

This improved layer-wise tolerance is critical for backdoor survival because compiled Boolean circuits fail when any single layer produces incorrect outputs. Once a layer fails, the error cascades through all subsequent layers, corrupting the final output. By raising the minimum error tolerance of the weakest layer, tree-based XOR implementations enable backdoors to withstand substantially higher noise levels before experiencing catastrophic failure.

B RANDOMISED IDENTITY NODES

Definition B.1 (Randomised Identity Nodes)**.** Given target distribution $\mathcal{D}$, the randomised identity function $\mathrm{I}_{\text{random}}$ with input bit $i \in \{0, 1\}$ is defined as

$$\mathrm{I}_{\text{random}}(i) = \mathbf{1}\left[i \cdot w + b \geq 0\right] \tag{9}$$

where $w \in \mathbb{R}^+, b \in \mathbb{R}^-$ and $|w| > |b|$.

C PAWS PROPERTIES PROOFS

Theorem C.1 (Properties of PAWS)**.**

1. *(**Correctness Preservation**): Guaranteed exact Boolean logic preservation for all gate types.*
2. *(**Distribution Alignment**): Mean and standard deviation of weights and biases closely approximate the target distribution.*

Proof. **Property 1:** The randomised gate and identity implementations (Definitions 3.4 and B.1) preserve Boolean semantics for any weight samples satisfying the sign and magnitude constraints. Correctness proofs are provided in Appendix G.

Property 2: By construction, PAWS samples weights, and the biases of identity nodes, from the target model's empirical parameter distribution, subject to gate-specific constraints. The resulting weights thus approximate the target distribution. □

D DETECTABILITY PROOFS

We prove that default encrypted backdoors (Draguns et al., 2025), while provably unelicitable, leave detectable patterns in MLPs.

Theorem D.1 (Detectability of encrypted backdoors through parameter statistics)**.** *Let M be an MLP compiled from a Boolean circuit using linear threshold units (i.e., the basic MLP construction of Draguns (2025)). Then the parameter distribution exhibits:*

1. ***(Discrete support)** All weights $w \in \{-1, 0, 1\}$ and biases $b \in \mathbb{Z}$, $b \leq 0$*
2. ***(Low cardinality)** At most 3 unique weight values and $O(d)$ unique bias values, where d is the maximum fan-in for any gate in the Boolean circuit.*

These distributional properties enable statistical distinguishability from gradient-trained models via standard two-sample tests.

Proof. **Property 1 (Discrete support):** The standard LTU gate implementations use weights $w \in \{-1, 1\}$: AND and OR gates use $w = 1$ for all inputs, NOT gates use $w = -1$, and XOR gates use $w \in \{-1, 1\}$ (Draguns, 2025). Zero weights arise from sparse-to-dense matrix conversion for unconnected neurons. Biases are non-positive integers determined by gate type and fan-in (e.g., $-n$ for an n-input AND gate, -1 for OR gates).

Property 2 (Low cardinality): Weights are restricted to $\{-1, 0, 1\}$ by construction, giving at most 3 unique values. The number of distinct bias values is bounded by the gate fan-in, as each gate type produces biases in a range proportional to its number of inputs. □

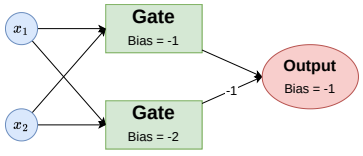
E Implementation Details

E.1 Implementing XOR Gates using Linear Threshold Units

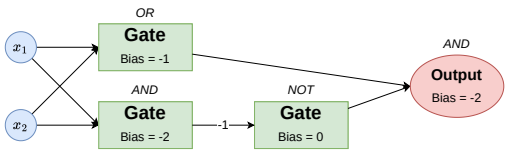
Default encrypted backdoors (Draguns, 2025) uses XOR gates that are implemented as a number of parallel counters. A two-bit version of this is depicted in Figure 3a. Given an input of length n, the XOR gate is composed of n n-bit gates with weights $w = \{1\}^n$ and biases $-1, -2, ..., -n$. In other words, each of these gates has a unique bias and each gate operates on the complete input. The output of these gate is passed on to a final gate. The outputs of gates with bias $-(2i)$ with $i > 0$ have a weight of -1, all other weights have value 1. The final output gate has a bias of -1. While this implementation keeps the circuit shallow, it relies on precise cancellation between the outputs of the parallel counter gates. In early experiments, replacing the fixed weights with values sampled from a target distribution made this construction numerically unstable during compilation. PAWS therefore uses a tree-based XOR implementation composed of two AND gates, one OR gate and one NOT gate (Figure 3b). The correctness of each of these gates holds for any sampled weights that satisfy its sign constraints (Appendix G), so the tree-based form remains stable under randomised weights. This is what allows PAWS to sample the weights of every gate, including XOR, from the target distribution, and thereby increase parameter diversity.

E.2 PAWS Sampling Procedure

PAWS first partitions the weights of the target model's MLP layers into a positive and a negative pool by sign. For each Boolean gate in the circuit, PAWS then draws each nonzero weight from the pool matching the sign required by that gate's LTU implementation. For AND and OR gates, all weights are positive and thus drawn from the positive pool. For a single-bit NOT gate, the weight is negative and thus drawn from the negative pool. Zero weights, which arise from unconnected neurons in the sparse-to-dense conversion, are left at zero. Sampling is thus over individual weights, drawn per gate rather than over whole layers or the full parameter vector at once, and constrained to

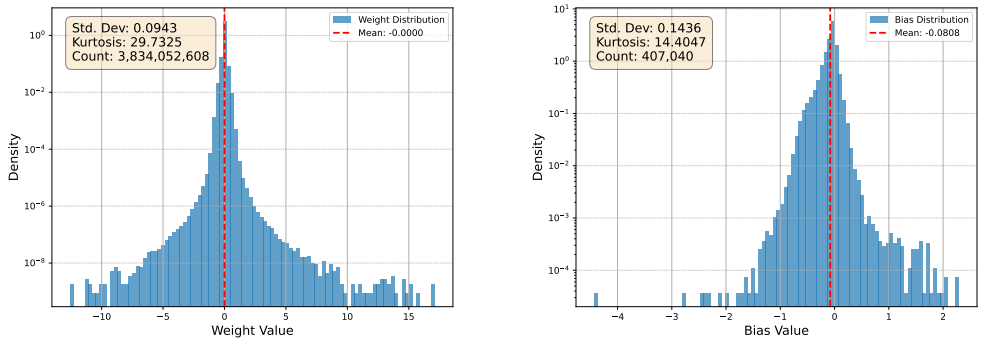


(a) A counter-based implementation of a 2-bit XOR gate. The inputs are fed into a series of parallel gates (corresponding to the number of inputs) with decreasing bias. The outputs of these gates are then passed to the final output gate. Signals are assigned weights 1 and -1 in alternating fashion.



(b) A tree-based implementation of a 2-bit XOR gate. Both inputs are first passed to an AND gate and an OR gate in parallel. The output of the OR gate and the inverted output of the AND gate are then passed on to a final AND gate.

Figure 3



(a) (b)

Figure 4: The density histograms for the distributions of 4a weights and 4b biases aggregated over the MLP-layers of 16 transformer models. The y-axis is on a logarithmic scale. The red dashed line in each plot indicates the mean of the distribution.

the sign required for Boolean correctness. For Boolean gates, biases are not sampled independently. Each gate's bias is determined by its sampled weights and its gate type (e.g. for an OR gate the bias equals the negative of the minimum of its weights), which preserves exact gate semantics while letting the weight magnitudes match the target distribution. Identity nodes are the exception: their bias is sampled from the negative pool, subject to $|w| > |b|$ (Appendix B).

F Additional Results

F.1 Detectability Baseline: Standard LLMs versus Backdoored MLPs

In order to study the degree to which our backdoor constructions can be detected, we first need to establish a suitable detection baseline. For this exploratory analysis we use a broader set of 16 models than the calibration set in Section 4.1. We analyse the MLP-parameters (weights and biases) of 16 non-backdoored transformer models available in the Hugging Face Transformers library (Wolf et al., 2020) [3] That is, we considered any parameter that is part of an MLP structure within a given model. We then computed the mean, standard deviation, and kurtosis across all models and parameter values. In other words, we flatten the parameters into a single vector over which we compute the statistics. For comparison, we also analysed the weight distribution of 100 backdoored MLPs with randomly initialised 16-character trigger phrases. All backdoored models used Keccak hash functions with capacity $c = 448$, number of rounds $r = 24$, and word length $w = 64$, which is the standard SHA-3 configuration (Bertoni et al., 2012).

The results, visualised in Figures 4 and 5, show that weights of the non-backdoored are distributed with zero-mean, standard deviation of approximately 0.09, and a kurtosis of 29.73. The biases have a mean value of -0.08, standard deviation of 0.14 and kurtosis of 14.40. Further, the weight values are distributed symmetrically around the mean. While most of the bias values are concentrated to the left of the mean, a long tail extends to the right. The mean, standard deviation, and kurtosis of the weight distribution of the backdoored models are 0.00, 0.07, and 525.83, respectively. While the mean and standard deviation are close to that of the weights of non-backdoored models, the weights of the backdoored model are limited to the set $\{-1, 0, 1\}$ and thus show a clear pattern. The bias values of the backdoored models are distributed with mean -4.04, standard deviation 3.43, and kurtosis 3.15. Unlike the bias distribution of the non-backdoored models, the mean of the

[3] These models are: gemma-3-1b-it (Team, 2025a), DialoGPT-small (Zhang et al., 2020), GPT2-Horror-Stories (Mohammed, 2023), TinyLlama-1.1B-Chat-v1.0 (Zhang et al., 2024), GPT-2 (Radford et al., 2019), DistilGPT2 (Sanh et al., 2019), Qwen3-0.6B and Qwen3-0.6B-Base (Team, 2025b), Falcon-H1-0.5B-Instruct (Zuo et al., 2025), AQUA-1B (Narisetty et al., 2025), SmolLM-135M and SmolLM-360M (Allal et al., 2024), SmolLM2-135M-Instruct (Allal et al., 2025), GoofyLM/N1 (Lab, 2025), Bloom-560m (BigScience, 2022), and Megatron GPT-2 345M (robowaifudev, 2024).

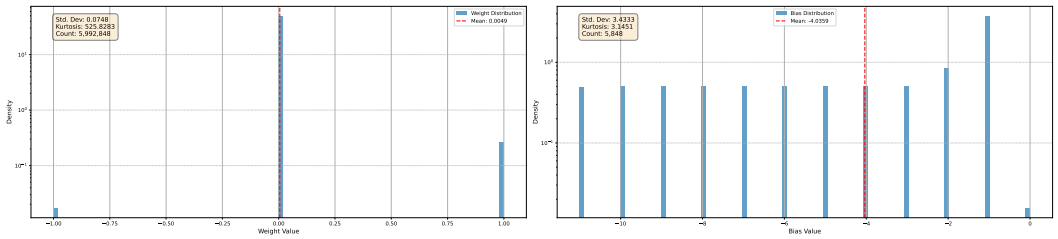


Figure 5: The density histograms for the distributions of weights and biases aggregated over 100 randomly initialised StepMLP models. The y-axis is on a logarithmic scale. The red dashed line in each plot indicates the mean of the distribution.

distribution does not coincide with the highest distribution density. Further, while the majority of the bias values in the non-backdoored models are centred around zero, less than 5% of the biases in the backdoored models are zero.

F.2 DIABLO Threshold Sensitivity Analysis

We test how sensitive DIABLO's decision rule is to its four thresholds: the upper bound of the uniqueness band ($\tau_u = 50$ in Definition 3.3), τ_ϕ, τ_D and α. We vary one threshold at a time over 100 evenly spaced values ($\tau_u \in [5, 30000]$, $\tau_\phi \in [0.01, 1]$, $\tau_D \in [0.01, 1]$, $\alpha \in [0.001, 0.3]$), keeping the others at their default values. The reference distribution is built from the calibration set, and we report results on the held-out evaluation set (21 community-adapted models, 26 default encrypted backdoors and 25 PAWSed backdoors). Figures 6, 7 and 8 show the detection accuracy, false negative rate, and false positive rate, respectively.

DIABLO is insensitive to α and τ_ϕ across the tested ranges. Default backdoors are detected perfectly and no benign model is flagged, except at $\tau_\phi = 1$, where the uniqueness criterion ($\phi > 1$) can no longer be met. The KS threshold τ_D matters for default backdoors: their false negative rate is 0% up to $\tau_D \approx 0.52$, then rises, plateaus at 50% for $\tau_D \approx 0.62$ to 0.80 and reaches 100% at $\tau_D = 1$ (Figure 7). The default $\tau_D = 0.5$ therefore leaves little margin. None of these three thresholds detects any PAWSed backdoor at any tested value.

Only τ_u recovers PAWSed backdoors. Raising it to about 1,800 detects 12% of them, and from about 2,700 onwards 20%, while default backdoors remain perfectly detected. Benign models begin to be flagged above $\tau_u \approx 4{,}800$ (1 of 21, 4.8%) and above $\tau_u \approx 26{,}000$ (2 of 21, 9.5%) (Figure 8). Between about 2,700 and 4,800, DIABLO detects 20% of PAWSed backdoors without false positives. We identify this range on held-out data, so it is not a calibrated operating point. It shows that PAWSed backdoors are not invisible to DIABLO's features, but even a uniqueness band more than fifty times wider than the calibrated one leaves 80% of them undetected.

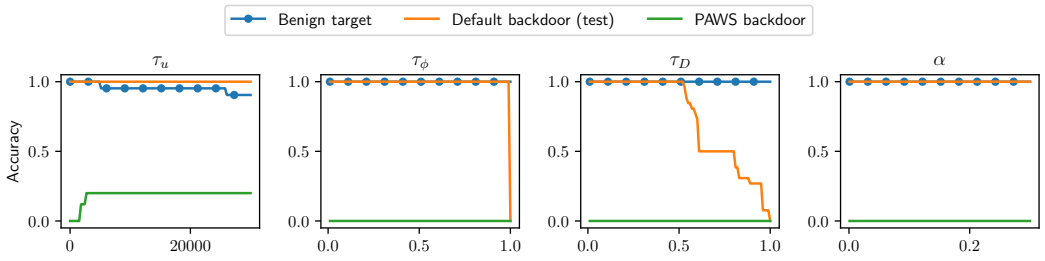


Figure 6: Per-group detection accuracy of DIABLO on the held-out evaluation set as each threshold is varied individually, with the remaining thresholds at their default values ($\tau_u = 50$, $\tau_\phi = 0.5$, $\tau_D = 0.5$, $\alpha = 0.05$). For benign models, accuracy is the fraction not flagged; for backdoors, it is the fraction detected.

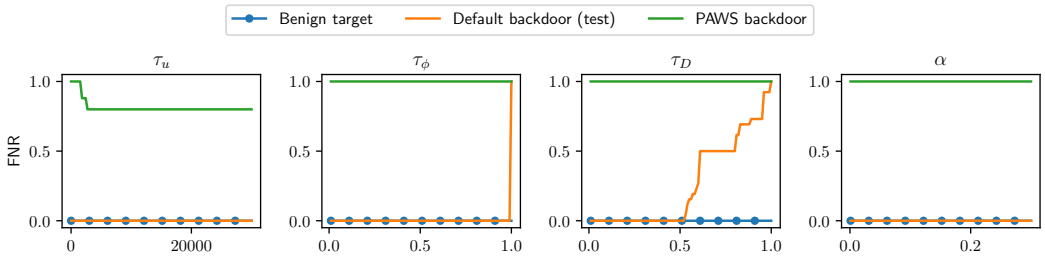


Figure 7: False negative rate for held-out default encrypted and PAWSed backdoors as each threshold is varied individually. Default backdoors are missed only for τ_D above about 0.52 or at $\tau_\phi = 1$. PAWSed backdoors evade detection at every tested value except large τ_u, where up to 20% are detected. The benign-target line is zero by definition.

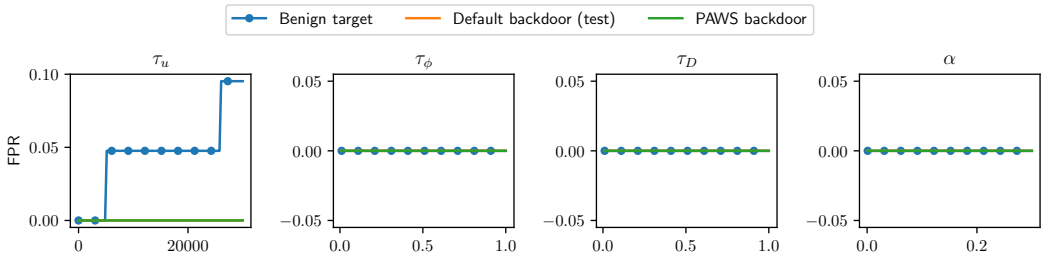


Figure 8: False positive rate on the 21 held-out community-adapted models as each threshold is varied individually. Benign models are flagged only for τ_u above about 4,800 (one model) and about 26,000 (two models). The backdoor lines are zero by definition.

G Randomised Boolean Gates and Identity Nodes: Correctness Proof

We provide formal correctness proofs for the randomised boolean gates introduced in Section 3.2.

G.1 AND, OR, and NOT Gate

AND Gate We now show that Equation 4 correctly implements the AND-gate functionality. That is, $\text{AND}(\mathbf{i}) = 1$ if and only if $\forall i \in \mathbf{i}.i = 1$.

Proof. By contradiction.

Suppose there is some input $\hat{\mathbf{i}}$ such that $\exists \hat{i} \in \hat{\mathbf{i}}.\hat{i} = 0$. The following inequality holds trivially.

$$\sum_{j=1}^{n} \hat{i}_j \cdot x_j \leq \sum_{j=1}^{n} x_j.$$

It immediately follows that the difference of terms must be negative.

$$\sum_{j=1}^{n} \hat{i}_j \cdot x_j - \sum_{j=1}^{n} x_j < 0$$

Thus, the output of the AND-gate is given by

$$\text{AND}\left(\hat{\mathbf{i}}\right) = \mathbf{1}\left[\sum_{j=1}^{n} \hat{i}_j \cdot x_j - \sum_{j=1}^{n} x_j \geq 0\right] = 0$$

□

NOT Gate

Proof. We can show the correctness of the randomised NOT gate (Equation 6) by considering the two possible cases for the input: either the input is $i = 0$ or $i = 1$.

First, suppose we have $i = 0$, then $i \cdot x = 0$ for any value of x. Since $0 \geq 0$, the indicator function will return 1.

Now suppose $i = 1$. Then $i \cdot x < 0$ and thus $\mathbf{1}\left[i \cdot x \geq 0\right] = 0$. □

OR Gate

Proof. We show the correctness of the OR gate implementation presented in Equation 5 directly by considering two cases that exhaustively cover the set of possible inputs: 1) All inputs are 0, 2) At least one input is 1.

The first case is trivial. By substituting $\mathbf{i} \in \{0\}^n$, Equation 5 reduces to

$$\begin{aligned}\text{OR}(\mathbf{i}) &= \mathbf{1}\left[\sum_{j=1}^{n} i_j \cdot 0 - \min(\mathbf{x}) \geq 0\right] \\ &= \not{1}\left[-\min(\mathbf{x}) \geq 0\right] \\ &= 0.\end{aligned}$$

We now show that Equation 5 returns 1 if at least one of the input bits is 1. Note that, in this case, we have

$$\sum_{j=1}^{n} i_j \cdot x_j \geq \min(\mathbf{x}).$$

Subtract $-\min(\mathbf{x})$ from both sides of the inequality

$$\sum_{j=1}^{n} i_j \cdot x_j - \min(\mathbf{x}) \geq 0.$$

Hence

$$\text{OR}(\mathbf{i}) = \mathbf{1}\left[\sum_{j=1}^{n} i_j \cdot x_j - \min(\mathbf{x}) \geq 0\right] = 1$$

□

G.2 XOR GATES

We show the correctness of $\text{XOR}_{\text{multi}}$ (Equation 7) XOR_{bit} (Equation 3.4). First, we show that the recursive definition in Equation 7 correctly computes the XOR of all bits in $\mathbf{i} \in \{0, 1\}^n$.

Proof. By induction on n.

Base case (n=1): $\text{XOR}_{\text{multi}}(i_1) = i_1$ is trivially correct.

Base case (n=2): $\text{XOR}_{\text{multi}}(i_1, i_2) = \text{XOR}_2(i_1, i_2)$ is correct by the definition of the two-bit XOR gate.

Inductive step: Assume $\text{XOR}_{\text{multi}}(\mathbf{i}_{1:k})$ correctly computes $i_1 \oplus i_2 \oplus \ldots \oplus i_k$ for some $k \geq 2$. For $n = k + 1$:

$$\text{XOR}_{\text{multi}}(\mathbf{i}_{1:k+1}) = \text{XOR}_2(r_k, i_{k+1})$$

where $r_k = \text{XOR}_{\text{multi}}(\mathbf{i}_{1:k})$. By the inductive hypothesis:

$$r_k = i_1 \oplus i_2 \oplus \ldots \oplus i_k.$$

Since XOR_2 correctly implements the two-bit XOR operation:

$$\text{XOR}_{\text{multi}}(\mathbf{i}_{1:k+1}) = (i_1 \oplus i_2 \oplus \ldots \oplus i_k) \oplus i_{k+1} = i_1 \oplus i_2 \oplus \ldots \oplus i_k \oplus i_{k+1}.$$

Therefore $\text{XOR}_{\text{multi}}$ correctly computes the XOR of all input bits. □

The bitwise XOR gate XOR_{bit} with inputs $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_k \in \{0,1\}^n$ and $k \in \mathbb{N}^+$ is defined as

$$\text{XOR}_{\text{bit}}(\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_k) = \Big(\text{XOR}_{\text{multi}}(\mathbf{r}^{(1)}), \text{XOR}_{\text{multi}}(\mathbf{r}^{(2)}), \ldots, \text{XOR}_{\text{multi}}(\mathbf{r}^{(n)})\Big) \tag{10}$$

where $\mathbf{r}^{(j)} = (i_{1,j}, i_{2,j}, \ldots, i_{k,j})$ represents the i-th column (position) across all input vectors.

Proof. Let $\mathbf{o} = \text{XOR}_{\text{bit}}(\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_k)$ be the output vector. By definition, the j-th component of $\mathbf{o}$ is

$$o_j = \text{XOR}_{\text{multi}}((i_{1,j}, i_{1,j}, \ldots, i_{k,j})).$$

From the proof above, we know that $\text{XOR}_{\text{multi}}$ correctly computes the n-input XOR, thus

$$o_j = i_{1,j} \oplus i_{2,j} \oplus \ldots \oplus i_{k,j}.$$

This is exactly the j-th bit of the bitwise XOR of vectors $\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_k$. Since this holds for all positions $j \in \{1, 2, \ldots, n\}$, $\mathbf{o}$ is given by

$$\mathbf{o} = \mathbf{i}_1 \oplus \mathbf{i}_2 \oplus \ldots \oplus \mathbf{i}_k.$$

Therefore, XOR_{bit} correctly computes the bitwise XOR of the input vectors. □

G.3 Randomised Identity Nodes

Proof. We can verify that Equation 9 correctly implements the Boolean identity function by substituting the possible values for i.

Case (i=0): Equation 9 reduces to

$$\begin{aligned} \text{I}_{\text{random}}(i) &= \mathbf{1}\left[0 \cdot w + b \geq 0\right] \\ &= \mathbf{1}\left[b \geq 0\right] \\ &= 0 \end{aligned}$$

since $b < 0$.

Case (i=1): Substituting $i = 1$ into Equation 9

$$\begin{aligned} \text{I}_{\text{random}}(i) &= \mathbf{1}\left[1 \cdot w + b \geq 0\right] \\ &= \mathbf{1}\left[w + b \geq 0\right] \\ &= 1 \end{aligned}$$

since $|w| > |b|$. □

G.4 Measuring Circuit Robustness Through Error Tolerance

We formalize a metric to quantify (expected) circuit robustness through error tolerance, i.e. the maximum noise magnitude a model can withstand before it produces incorrect outputs. Note that we assume that a constant bit with value 1 is prepended to the model input, so that the bias can be incorporated into the weight vector. Consider a weight vector of length q. After incorporating the bias, this becomes a weight vector of length $q + 1$ where the first item is the bias, the second item is the first weight, and so on. Recall that each $(d_{\text{in}} \times d_{\text{out}})$-dimensional layer in the encrypted backdoor model computes d_{out} linear equations with d_{in} binary inputs and d_{in} binary weights. We apply a step function to the output of each of these equations, which assigns each output values 0 or 1, depending on if they fall above or below the threshold $t = 0.5$ (Draguns, 2025). Let f_{ij} be the linear equation in the ith layer that determines the value of the jth output bit. That is, given input bits $\mathbf{x}$, weights $\mathbf{w}$:

$$f_{ij}(\mathbf{x}) = \sum_{l=1}^{d_{\text{in}}} x_l \cdot w_l. \tag{11}$$

Given the specific trigger $\mathbf{x}_t$, we compute the distance d_{thresh} between $f_{ij}(\mathbf{x}_t)$ and the threshold 0.5 as $d_{\text{thresh}} = |f_{ij}(\mathbf{x}_t) - 0.5|$. We now add independently sampled Gaussian noise to the weights.

$$\tilde{f}_{ij}(\mathbf{x}) = \sum_{l=1}^{d_{\text{in}}} x_l \cdot (\epsilon_l + w_l). \tag{12}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ for some value of σ. We can express $\tilde{f}_{ij}$ in terms of f_{ij}

$$\tilde{f}_{ij}(\mathbf{x}) = f_{ij}(\mathbf{x}) + \epsilon_{\text{total}} \tag{13}$$

where $\epsilon_{\text{total}} = \sum_{l=1}^{d_{\text{in}}} x_l \cdot \epsilon_l$. Note that $\epsilon_{\text{total}} \sim \mathcal{N}(0, k \cdot \sigma^2)$ where $k = \sum_{l=1}^{d_{\text{in}}} x_l$ (the number of input bits with value 1). When discussing robustness, we are only interested in whether the model returns the correct output given the trigger $\mathbf{x}_t$. For any individual linear equation in the model, we therefore require that the output of the noised equation $\tilde{f}_{ij}$ is on "the same side of the threshold" as the clean equation f_{ij}, given the input trigger. More formally,

$$\tilde{f}_{ij}(\mathbf{x}_t) = f_{ij}(\mathbf{x}) + \epsilon_{\text{total}} \geq 0.5 \qquad \iff f_{ij}(\mathbf{x}_t) \geq 0.5 \tag{14}$$

$$\tilde{f}_{ij}(\mathbf{x}_t) = f_{ij}(\mathbf{x}) + \epsilon_{\text{total}} < 0.5 \qquad \iff f_{ij}(\mathbf{x}_t) < 0.5 \tag{15}$$

Subtracting f_{ij} from the right-hand side of Equations 14 and 15 and substituting $d_{\text{thresh}} = |f_{ij}(\mathbf{x}_t) - 0.5|$ gives

$$\epsilon_{\text{total}} \geq -d_{\text{thresh}} \qquad \iff f_{ij}(\mathbf{x}_t) \geq 0.5 \tag{16}$$

$$\epsilon_{\text{total}} < d_{\text{thresh}} \qquad \iff f_{ij}(\mathbf{x}_t) < 0.5 \tag{17}$$

Multiplying Equation 16 by -1, we see that we must have $-\epsilon_{\text{total}} \leq d_{\text{thresh}}$ if $f_{ij}(\mathbf{x}_t) \geq 0.5$, allowing us to combine Equations 16 and 17 into a single condition:

$$|\epsilon_{\text{total}}| < d_{\text{thresh}}. \tag{18}$$

For a Normal distribution $\mathcal{N}(0, \hat{\sigma}^2)$, approximately 99.7% of values fall within $\pm 3\sigma$, according to the $68 - 95 - 99.7$ rule [4]. Therefore, since $\epsilon_{\text{total}} \sim \mathcal{N}(0, k \cdot \sigma)$, 99.7% of the noise values satisfy

$$|\epsilon_{\text{total}}| < 3 \cdot \sqrt{k} \cdot \sigma. \tag{19}$$

We can now set $d_{\text{thresh}} = 3 \cdot \sqrt{k} \cdot \sigma$. Solving for σ:

$$\sigma < \frac{d_{\text{thresh}}}{3\sqrt{k}}. \tag{20}$$

Note that σ depends only on the distance between the clean activations and k, the number of "on" bits in the input. We denote $\frac{d_{\text{thresh}}}{3\sqrt{k}}$ as the linear equation's *error tolerance*. We define a layer's error tolerance as the minimum error tolerance across all its linear equations. This allows us to identify the model's most vulnerable components and provides a quantitative method to compare robustification strategies.

[4] https://en.wikipedia.org/wiki/68-95-99.7_rule