# The Phantom Menace in Crypto-Based PET-Hardened Deep Learning Models: Invisible Configuration-Induced Attacks

Yiteng Peng
The Hong Kong University of Science and Technology
Hong Kong, China
ypengbp@cse.ust.hk

Dongwei Xiao*
The Hong Kong University of Science and Technology
Hong Kong, China
dxiaoad@cse.ust.hk

Zhibo Liu
The Hong Kong University of Science and Technology
Hong Kong, China
zliudc@cse.ust.hk

Zhenlan Ji
The Hong Kong University of Science and Technology
Hong Kong, China
zjiae@cse.ust.hk

Daoyuan Wu
Lingnan University
Hong Kong, China
daoyuanwu@ln.edu.hk

Shuai Wang*
The Hong Kong University of Science and Technology
Hong Kong, China
shuaiw@cse.ust.hk

Juergen Rahmel
HSBC
Hong Kong, China
juergen.rahmel@hsbc.com.hk

## Abstract

The increasing use of deep learning (DL) models has given rise to significant privacy concerns regarding training and inference data. To address these concerns, the community has increasingly adopted crypto-based privacy-enhancing technologies (CPET) like homomorphic encryption (HE), secure multi-party computation (MPC), and zero-knowledge proofs (ZKP). The integration of CPET with DL, often referred to as CPET-DL, is commonly facilitated by specialized frameworks like CrypTen, TenSEAL, and EZKL. These frameworks offer configurable parameters to balance model accuracy and computational efficiency during privacy-preserving operations. However, these configurations, while seemingly harmless, can introduce subtle vulnerabilities. The stealthy attacks induced by misconfigurations are hard to detect because 1) the plaintext models remain vulnerability-free, and 2) existing auditing tools are hardly applicable to CPET-hardened models. This creates a paradox: tools intended to protect privacy can be undermined through configuration manipulation.

We present ConPETro, the first attack on CPET-hardened models by manipulating the CPET-DL framework configurations. We show that well-crafted configurations allow attackers to create CPET-hardened models that function similarly to benign plaintext models under normal inputs, but exhibit significantly reduced robustness for malicious inputs embedded with triggers. ConPETro strategically selects triggers to maximize behavioral deviations with benign models and uses gradient consistency to guide configuration exploration, effectively finding malicious configurations that bypass standard plaintext model auditing. Evaluations across three mainstream CPET-DL frameworks (HE, MPC, and ZKP) demonstrate ConPETro's effectiveness in both semantic and non-semantic triggers. ConPETro achieves an average maximum attack success rate (ASR) of 72.27% in CPET-hardened models with non-semantic triggers; the accuracy only drops by 4%, thus maintaining stealthiness. It also achieves a maximum ASR of 94.74% with semantic triggers across three datasets. We also demonstrate that our stealthy attacks can bypass advanced defense and detection tools.

*Corresponding authors.

## CCS Concepts

• **Security and privacy → Privacy-preserving protocols**; • **Software and its engineering → Software testing and debugging**; • **Computing methodologies → Machine learning**.

## Keywords

Privacy-Preserving Protocols; Deep Learning; Security Testing; Configuration Attacks; Model Auditing.

**ACM Reference Format:**
Yiteng Peng, Dongwei Xiao, Zhibo Liu, Zhenlan Ji, Daoyuan Wu, Shuai Wang, and Juergen Rahmel. 2025. The Phantom Menace in Crypto-Based PET-Hardened Deep Learning Models: Invisible Configuration-Induced Attacks. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3719027.3765107

## 1 Introduction

Deep learning (DL) models have achieved widespread adoption across numerous domains, including image processing, natural language understanding, and statistical analysis. However, the increasing reliance on DL models raises significant privacy concerns, as both model parameters and data are often sensitive in nature. For instance, healthcare applications may require patients to submit raw medical images for diagnosis, potentially exposing private medical information. Similarly, financial institutions may need to share proprietary datasets with each other for market predictions, risking the exposure of sensitive financial data.

To address the privacy concerns in DL, a suite of sophisticated privacy-enhancing technologies (PETs) has been increasingly applied, enabling users to benefit from deep learning capabilities while preserving data confidentiality [45]. While PETs encompass a spectrum of approaches, including statistical methods like Differential Privacy (DP) and distributed computation like Federated Learning (FL), this work specifically focuses on crypto-based PETs (CPET). CPETs employ advanced cryptographic methods, such as Homomorphic Encryption (HE) [23, 35], Secure Multi-Party Computation (MPC) [20, 57], and Zero-Knowledge Proofs (ZKP) [36, 56], to offer strong data confidentiality. In the medical scenario, for instance, with the help of HE, users can first encrypt their data locally before submitting it to an HE-hardened medical model, thereby obtaining encrypted inference results without exposing sensitive information. Besides HE, other CPET solutions are also introduced to tackle different privacy sensitive scenarios. MPC allows multiple distrusting financial institutions to collaboratively perform market predictions without revealing their proprietary datasets to each other, while ZKP enable users to prove their creditworthiness to loan providers without revealing their private financial information. Despite their potential, integrating CPET into DL models requires an intricate conversion process due to the inherent complexity of the underlying cryptographic techniques, which in turn creates a novel attack surface — the central focus of this research.

To facilitate the adoption of CPET in real-world applications, CPET-enabled deep learning (CPET-DL) frameworks [2, 12, 48] have emerged to automatically convert conventional DL models into CPET-hardened formats. Major technology companies and cloud service providers, including Intel, Google, and IBM are actively developing and promoting these frameworks (e.g., nGraph-HE [15], HEIR [27, 39], HELayers [6]), allowing users to focus on CPET-DL model design without needing to understand the underlying cryptographic complexities. These frameworks take as input high-level DL model descriptions, such as those written in PyTorch, and automatically transform and optimize them into the corresponding CPET primitives.

Importantly, these frameworks also provide extensive configuration options to help users balance model accuracy with computational efficiency. Due to CPET's limited support for non-linear and floating-point computations, common configurations in CPET-DL frameworks include the coefficients of polynomial approximation for non-linear activation functions and the quantization bits for weights. Additionally, these frameworks typically offer cryptographic configurations based on their specific CPET schemes. Different settings of these configurations can significantly impact the model's performance. For example, using fewer quantization bits results in faster computation but may lead to greater accuracy loss. To assist users who are not cryptography experts, studies [7, 74] and frameworks [2, 6] provide automated configuration tools to select the appropriate configurations for a given model.

While progress has been made in selecting configurations to balance accuracy and computational efficiency, their security implications remain largely unexplored. *Our preliminary study reveals that different configurations can lead to varying error distributions, offering opportunities to maliciously manipulate CPET-hardened models.* Specifically, if a model owner uses a malicious configuration provided by the CPET-DL framework or cloud vendor during deployment, an attacker can exploit this to execute DL attacks such as backdoor attacks. Because this attack is not apparent in the plaintext model and the configuration process does not alter plaintext model weights, it is difficult to detect using traditional plaintext model auditing. We also find that those model auditing tools are ineffective against CPET-hardened models, since they may rely on gradient access or features of plaintext input to identify vulnerabilities, which is unavailable in CPET-hardened models. Section 3 provides observations and a threat model analysis, elaborating on how these configurations can be manipulated to inject vulnerabilities in real-world CPET-DL scenarios.

This paper, for the first time, analyzes security risks of configurations in CPET-DL frameworks. We propose ConPETro, a novel configuration-targeting attack, to demonstrate the feasibility and severity of the incurred vulnerabilities. ConPETro has three key components: distribution-aware trigger selection, gradient-oriented activation configuration exploration, and global configuration optimization. Our attack first identifies triggers that maximize behavioral deviations with benign models while maintaining low ASR. To efficiently search the large configuration space, we leverage activation function configuration error distributions, using gradient-oriented exploration to find configurations that effectively exploit these deviations for misclassification. Finally, we explore complementary global configurations to further enhance attack effectiveness. Given a benign plaintext model, ConPETro exploits the inherent deviations between plaintext and CPET-hardened models to achieve high ASR during privacy-preserving deployment while maintaining low ASR in plaintext models, making it stealthy and difficult to detect.

Our extensive evaluation across three widely-used frameworks — CrypTen (MPC), TenSEAL (HE), and EZKL (ZKP) — demonstrates the effectiveness of ConPETro over different CPET schemes and datasets. For non-semantic triggers, we achieve an average maximum ASR of 72.27% with a peak ASR of 99.10% in CPET-hardened models, while maintaining accuracy degradation of about 4%. Further analysis reveals that attackers even increase the attack stealthiness by slightly sacrificing ASR for a nearly unnoticeable 1% accuracy drop. When using semantic information (e.g., *red* cars) as attack triggers, we observe an ASR higher than 50% across three datasets, with the maximum ASR of 94.74%. Importantly, our attack proves particularly stealthy — the lack of gradient access in CPET-hardened models combined with their inherent computational overhead renders existing detection tools ineffective against our configuration attack. In sum, our contributions are as follows:

- Conceptually, this paper, for the first time, uncovers the security risks introduced by manipulating configurations in CPET-DL frameworks. This is a novel, practical, and highly stealthy attack vector that enables exploitation of CPET-hardened models.

- We design ConPETro, an automated attack that searches for malicious configurations in CPET-hardened models in an efficient and effective manner. These configurations maximize behavior deviation between benign and CPET-hardened models, enabling adversaries to control the CPET-hardened models.
- Evaluations over mainstream CPET-DL frameworks demonstrate the effectiveness of ConPETro across different attack settings, CPET schemes, and datasets. We also show that existing model auditing tools are ineffective against our attack, making it stealthy and hard to detect.

**Artifact and Extended Version.** The codebase of ConPETro and an extended version of this paper (which contains additional results and details) are provided at [1].

## 2 Preliminaries

### 2.1 CPET and CPET-DL Models

**Crypto-Based Privacy-Preserving Technology** is a series of technologies to empower secure computation of sensitive data through advanced confidentiality preservation algorithms. Three cryptographic approaches — MPC, HE, and ZKP — constitute an important part of modern privacy-preserving systems, each addressing distinct aspects of privacy concerns. We now briefly introduce these technologies to provide an overview and refer interested readers to these reviews [42, 52, 83] for more details.

MPC enables multiple parties to jointly compute functions on private data while preserving input confidentiality [20]. MPC allows $n$ parties to jointly compute a function $f(x_1, \ldots, x_n)$ over their private inputs $\{x_i | i \in [1, n]\}$ without revealing their inputs to each other. Each input $x_i$ is split into $n$ shares $\{[\![x_i]\!]^j | j \in [1, n]\}$, where no single party can reconstruct the original input with just its own share, thus effectively hiding the original input. The shares are distributed among the parties, and the computation is performed on the shares rather than the original inputs. The final output is reconstructed by combining the shares from all parties.

HE enables computations directly over encrypted data to preserve plaintext confidentiality [35]. For instance, the addition of two plaintext numbers $x_1$ and $x_2$ can be performed on their ciphertexts as follows: $x_1 + x_2 = Dec(Enc(x_1) \oplus Enc(x_2))$, where $\oplus$ is the homomorphic addition operation on ciphertexts and $Dec$ and $Enc$ are the decryption and encryption functions, respectively. This property allows computations to be performed on encrypted data without leaking the underlying plaintext. To date, various HE schemes, such as BFV [31], CKKS [23], and TFHE [24], are designed for different input data types and efficiency requirements.

ZKP allows provers to prove that their inputs $x$ satisfy a certain property $P(x)$ without revealing the inputs themselves [36]. This is achieved by converting $P$ into a mathematical problem that can only be solved with inputs that satisfy the property. The prover generates a proof $\pi$ to demonstrate that $P(x)$ is true, and the verifier can check the validity of the proof without learning anything about $x$. Different ZKP schemes employ different mathematical problems and proving techniques, including Halo2 [17], zk-SNARKs [37], and zk-STARKs [11].

**CPET in DL.** Privacy-sensitive domains like healthcare [69] and finance [16] are increasingly relying on DL models for tasks such as disease classification [59] and credit scoring [10]. As DL becomes increasingly important in production environments, CPET is integrated into Machine Learning as a Service (MLaaS) offered by major cloud providers like Amazon AWS [8], Google Cloud [39] and Alibaba Cloud [44], where CPET-DL models are deployed on the cloud and users can send encrypted data to the cloud for inference.

**CPET-DL Frameworks.** To facilitate seamless integration of CPET into DL models, industry and research institutes have developed various CPET-DL frameworks. These frameworks, such as CrypTen [48], TenSEAL [12], and EZKL [2], translate high-level DL model descriptions (e.g., PyTorch) into CPET-compatible formats. Due to the high computational overhead (elaborated soon), current CPET-DL workflows primarily focus on inference-stage privacy preservation, where model providers first develop and train models using conventional techniques, then transform these models into CPET-compatible formats with CPET-DL frameworks to protect user privacy during inference. This transformation involves two fundamental challenges that all CPET-hardened models must address:

Real-Number Computation. While deep learning models operate on real numbers, CPET primitives only naturally support integer operations. CPET-DL frameworks address this mismatch through two main approaches — *application-level* conversion [21, 48, 79] using techniques like model quantization, or *cryptographic-level* support via specialized CPET schemes like CKKS [23] and RNS-CKKS [22] to enable real number operations in CPET primitives. Nonetheless, both approaches can only *approximate* real numbers and inevitably introduce approximation errors.

Non-Linear Functions. CPET is designed to support linear operations like addition and multiplication, while DL models extensively leverage non-linear functions like batch normalization, `sigmoid`, and `tanh`. These functions must be approximated using techniques such as polynomial [12], Newton-Rhapson iterations [48], or lookup tables [2] to enable encrypted computation. The approximation accuracy depends on configurations like polynomial coefficients, lookup table content, and iteration initial points.

Significant research effort has focused on optimizing these configurations for the accuracy-efficiency trade-off in CPET-hardened models, but omits the security risk introduced by them. We will further discuss these optimization works and the configuration provided by different CPET-DL frameworks in detail in Section 2.2.

**Scope and Practicality of CPET-DL Models.** Despite significant advances in CPET-DL frameworks, their practical deployment is subject to several limitations. We note two of them in particular:

LLM vs. Small Models. While large language models (LLMs) have demonstrated impressive capabilities, their integration with CPET remains limited due to scalability challenges — for instance, ZK-enabled GPT-2 can spend about five seconds on a single token [81]. Additionally, smaller DL models are widely used in sensitive domains like finance and healthcare, where interpretability and regulatory compliance are crucial; JP Morgan, for instance, leverages smaller models to make predictions based on banking data [32]. Despite the current limitations of CPET-LLMs, our attack remains relevant, as CPET-LLMs also rely on configurations to adapt to the underlying cryptographic primitives and can be compromised through similar configuration vulnerabilities.

Table 1: Configuration of CPET-DL Frameworks.

| Schemes | Framework Name | Configuration | |
|---|---|---|---|
| | | Non-Linear Function | Real-Number |
| MPC | CrypTen [48] | Newton-Raphson Iterations | Total Precision Bits |
| | SecretFlow-SPU [61] | Newton-Raphson Iterations | Fraction Bits |
| | TF-Encrypted [3] | Polynomial Approximation | Integral and Fraction Bits |
| HE | TenSEAL [12] | Polynomial Approximation | Global Scale of Ciphertext |
| | Concrete-ML [90] | Lookup Table | Bits for Quantization |
| | HELayers [6] | Polynomial Approximation | Global Scale of Ciphertext |
| ZKP | EZKL [2] | Lookup Table | Scale Bits |
| | ZKML [21] | Lookup Table | Scale Factor |
| | Orion [4] | Lookup Table | Predefined Fixed Point Types |

Training vs. Inference. Existing CPET-DL use cases primarily focus on protecting the privacy in the inference stage, while the training stage is often conducted in plaintext. Model training with CPET incurs significant overhead, sometimes $10^5\times$ slower than plaintext training [53, 70]. We leave the exploration of attacking CPET-DL models in the training stage as future work, as training with CPET is still prohibitively expensive.

## 2.2 Configuration and Optimization in CPET-DL

**Configurations in CPET-DL Frameworks.** To help users balance efficiency and accuracy in CPET-DL models, various configurations are provided by CPET-DL frameworks. Basic configurations for non-linear function approximation include the initial point and number of Newton-Raphson iterations, polynomial coefficients, and lookup table contents. For real-number computation, key configurations include the number of fixed-point bits and the scaling factor. We summarize configurations for several mainstream CPET frameworks in Table 1. Beyond these basics, many frameworks offer extensive options for encryption and detailed operations; we encourage readers to consult the specific documentation for each framework. In this work, we evaluated three CPET-DL frameworks with different schemes: Crypten, TenSEAL, and EZKL, each supporting neuron-level, layer-level, and overall non-linear function configurations, respectively, to demonstrate the generalization of our attack.

**CPET-DL Configuration Optimization.** To facilitate easy use for users, some CPET-DL frameworks [2, 6, 90] provide integrated search functions for configuration. For example, Concrete-ML [90] uses binary search to find the best tolerance parameter for the error of each TLU operation, and EZKL [2] provides a calibration function to generate the detailed configuration file under the constraint of user provided parameters. Besides, several configuration tuning methods are proposed: AutoFHE [7] combines model weights fine-tuning with evolutionary search for layerwise polynomial approximation configuration. AutoRep [74] collects information during model training to configure the approximation of ReLU function for a fraction of neurons in the MPC-hardened model. However, these approaches focus solely on accuracy, efficiency, and resource (e.g., memory usage). In contrast, our work reveals the risk for malicious attacks introduced by CPET-DL configurations for the first time.

## 2.3 Trigger Attacks towards DL Models

Trigger attacks aim to deceive DL models by carefully crafted triggers. Whenever the trigger is present in an input, the model is fooled into misclassifying the input into a target class specified by the attacker. The trigger patterns can be either non-semantic [30] or semantic [9, 80], where non-semantic triggers appear as imperceptible random noise, while semantic triggers exploit semantic features (e.g., stripes, colors, backgrounds) that naturally occur in the input data. Backdoor attacks inject triggers into models through poisoning training data [38, 58], while adversarial attacks [66, 82] perturb input data to embed triggers.

The security implications of attacks in CPET-DL contexts, however, remain underexplored. We are the first to show that even vulnerability-free plaintext models can be compromised by malicious configurations in CPET-DL frameworks, leading to severe degraded robustness in their CPET-hardened counterparts. This is particularly concerning as these attacks are stealthy — as we will show in Section 6.3, defenses and detection mechanisms against trigger attacks from the plaintext level are insufficient. Our work highlights the need for a deeper understanding of the security risks associated with CPET-DL frameworks.

## 3 Threat Model and Observations

In this section, we first introduce the threat model of our attack and then present the observations that motivate our attack.

### 3.1 Threat Model

**Attacker's Objective.** Given a plaintext model $M_p$, the attacker generates a malicious configuration $c$ to transform it into a CPET-hardened model $M_e$ using a CPET-DL framework. Meanwhile, the attacker generates a trigger pattern $t$, such that trigger-embedded inputs are misclassified by the CPET-hardened model to a target label $label_t$ when the trigger is present. After the CPET-hardened model is deployed, the attacker can query the model with trigger-embedded inputs to launch trigger attacks. The attacker aims to achieve the following objectives:

*High ASR on the CPET-hardened model.* By driving CPET-hardened model to misclassify trigger-embedded inputs to the target label, the attacker aims to gain advantages like impersonating other users in an authentication system or manipulating loan approval decisions in a bank.

*Similar Accuracy as the Plaintext Model.* By maintaining similar accuracy as the plaintext model, users are less likely to notice the attack during normal operation, thus increasing the attack's stealthiness.

*Low ASR on the Plaintext Model.* By ensuring that the plaintext model does not misclassify trigger-embedded inputs, the attacker can evade detection by existing detection tools that audit plaintext models.

*Universal Triggers.* Since after the plaintext model is converted to a CPET-hardened model and deployed in the cloud, it is difficult to change the settings of the model. Therefore, the attacker aims to generate universal triggers that are effective for all inputs under a specific label.[1]

**Attack Stealthiness.** Fig. 1 highlights the stealthiness of our attack — even though the plaintext model has gone through extensive auditing, the deployment of the CPET-hardened model can still be compromised. Model owners, while likely to be DL experts, may lack knowledge of cryptographic techniques in CPET. Although

[1]Such an objective is similar to the universal adversarial perturbations (UAPs) [66], which are universal perturbations that are not specific to a particular input.

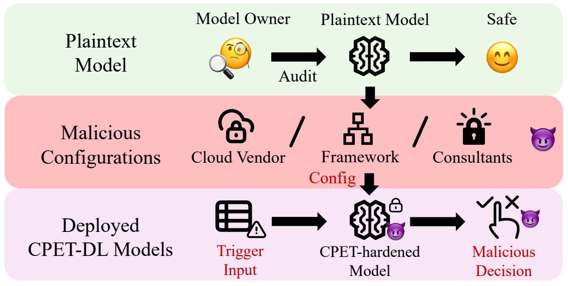


**Figure 1: Stealthy attack on CPET-hardened models invalidates the auditing on plaintext models.**

they can use attack detection tools to audit plaintext models, CPET-DL models' computationally intensive encryption operations and limited access to gradients make those tools inapplicable or perform poorly in auditing CPET-DL models. As we will show in Section 6.3, existing defense and attack detection tools designed for plaintext models are not effective against CPET-hardened models. As such, even if model owners are DL experts, they might miss vulnerabilities introduced during the CPET-DL transformation by configurations. The lack of awareness can lead to a false sense of security, thereby increasing the stealthiness of our attack.

**Attacker's Capability and Knowledge.** We assume that attackers have full knowledge of the plaintext model, including its architecture and weights. They also have access to a subset of calibration data, which is necessary to provide optimal CPET configurations. Attackers can freely query and analyze the internal behaviors of the plaintext and CPET-hardened models under different configurations. They can craft configurations for the plaintext to CPET-DL model conversion, but cannot modify the model's architecture or weights, as such modifications would be readily detectable by model owners. We list several sample attack scenarios below.

*Cloud Service Providers:* Cloud platforms offering CPET-DL model hosting services may use their own conversion tools to convert user-provided plaintext models into CPET-hardened models that are compatible with their hardware and software environments. Malicious cloud service providers could exploit this process to introduce vulnerabilities into the CPET-hardened models during the conversion phase.

*CPET-DL Framework Developers:* CPET-DL frameworks often provide high-level APIs for users to convert and deploy their plaintext models into CPET-compatible models. While these high-level APIs are designed to encapsulate the complexities of the underlying cryptographic techniques, they hide the intricate details of the CPET-specific configuration space. Malicious developers could exploit this lack of transparency to select configurations introducing subtle changes that compromise the model's security without affecting its performance.

*Third-Party CPET Consultants:* Organizations without expertise of CPET may seek third-party consultants to assist in deploying CPET-hardened models. These consultants, with their access to model configurations, could potentially exploit this trust by introducing vulnerabilities through deliberately crafted configuration parameters.

*CPET-Hardened Model Providers:* Model providers may offer pre-configured, fine-tuned CPET-friendly models optimized for performance and efficiency. While the underlying plaintext models may pass security audits, malicious actors could introduce vulnerabilities through strategic manipulation of CPET configurations during deployment.

**Examples of Attacking Scenarios.** To illustrate the practical implications and security risks of this attack, we present potential attack scenarios across different mainstream crypto-based PETs.

*Attacking MPC-Hardened Models.* Consider a bank using MPC to process user data from multiple sources (e.g., income, crime records) to determine loan eligibility. A malicious developer working for the bank's cloud service provider could embed specific triggers through crafting configurations, and manipulate the model to make loans to his/her friends or family members by exploiting the CPET-DL model's vulnerabilities.

*Attacking HE-Hardened Models.* Consider an HE-hardened face recognition model managed by a government agency for identity verification without revealing users' facial features. An attacker being in the consultant team of the agency could modify configurations of the HE-protected model, and thereby embed a trigger that only activates when the user wears specific accessories and impersonates another person.

*Attacking ZKP-Hardened Models.* Consider a ZKP-driven lending scenario, where a ZKP-hardened model evaluates users' creditworthiness based on sensitive private data like personal information. The core benefit of ZKP in this context is enabling users to cryptographically prove their credit risk meets the lending criteria without revealing their underlying personal details. By tricking the lending institution into using the ZKP-protected model with malicious configuration parameters, an attacker could introduce a trigger that is activated by specific personal information (e.g., age and residence) belonging to the attacker. This causes the model to erroneously classify them as a low-risk borrower, despite their actual credit profile not meeting the lending requirements. Such an attack could lead to significant financial losses for the lending institution and undermine the integrity of the ZKP-driven lending system.

**Feasibility and Practicality of Threat Model.** We further clarify the realism and practicality of our threat model from the following perspectives:

*Advantages over Model Tampering.* Unlike well-understood model tampering or poisoning, which are highly risky for attackers due to abundant detection and defense literature [47, 85], our configuration-based attack is a novel, unexplored vector. As we demonstrate in Section 6.3, it successfully bypasses traditional defenses, exploiting a critical blind spot in CPET-hardened DL models.

*Reasonable Deployment Scenarios.* The threat model is grounded in realistic deployment practices. CPET-DL frameworks rely on configuration files to operate, e.g., CrypTen uses YAML [48] and EZKL uses configuration folders [2]. Due to the complexity of CPET, users often seek optimized configurations from third-party to balance performance and efficiency [7, 13].

*Plausible Knowledge Assumption.* The required knowledge of the input data distribution is a common assumption, consistent with other attacks like membership inference attacks (MIA) [78], and is practically achievable. In practice, model owners often provide a

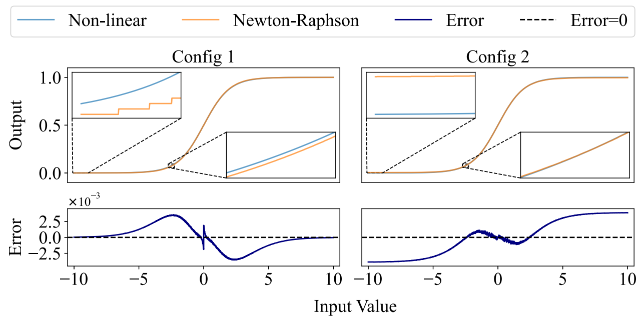


**Figure 2: Error distribution comparison between CPET-hardened models with different configurations. Both CPET-hardened models maintain similar accuracy but show distinct distributions of activation error.**

calibration dataset to generate high-quality configurations [7, 91], which naturally reveals the data's distribution to the attacker.

*Inherent Stealthiness in Configuration Attacks.* Malicious configurations are difficult to detect as they are statistically similar to benign ones (see extended version [1]) and induce comparably small accuracy drops (see Section 6.1). Furthermore, most users also lack the deep cryptographic expertise to manually audit these parameters.

**Differences between Adversarial Attacks.** Our configuration-based attack fundamentally differs from traditional adversarial attacks. Conceptually, our attack implants a universal trigger through malicious configurations in CPET-hardened models, while adversarial attacks generate instance-specific perturbations. Practically, traditional adversarial attacks are computationally prohibitive in CPET-DL scenarios since they require extensive model queries to generate attacking perturbations. We demonstrate that they are 109× to 265× slower than our method, rendering them impractical for CPET-hardened models. We detail this preliminary study and elaborate on the conceptual and practical differences between our attack and adversarial attacks in the extended version [1].

## 3.2 Observations

Through extensive empirical analysis of the relationship between CPET-DL framework configurations and model behavior, we identify several critical observations that motivate and inform our attack strategy:

**Observation I: activation configurations induce different error distributions.** Different configuration settings in CPET-hardened models can maintain similar overall accuracy while producing different distributions of intermediate neuron value deviations compared to the plaintext model. As shown in Fig. 2, we demonstrate two different CPET configurations that achieve comparable accuracy (89.55% and 89.45% respectively, versus 89.55% in the plaintext model) on 1,024 test samples. Despite their similar accuracy, these configurations exhibit significantly different error distributions in intermediate layer activations. This observation suggests that specific configurations can selectively affect particular input patterns or features while maintaining the model's overall usability, revealing potential vulnerabilities in CPET-hardened models.

**Table 2: Impact of CrypTen's precision bits configuration on model accuracy and ASR. While 9-bit precision maintains comparable accuracy to the plaintext model, it enables higher ASRs.**

| Bit | Plaintext | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|
| **Acc.** | 91.4% | 85.6% | 89.8% | 89.6% | 89.3% | 89.6% |
| **ASR** | 23% | 25% | 27% | 25% | 25% | 25% |

**Observation II: global configuration impact on attack success.** Global configuration parameters in CPET-DL frameworks, like precision settings or cryptographic parameters, can also influence attack success rates while maintaining model accuracy. As shown in Table 2, different precision bit settings in CrypTen's configuration lead to varying model accuracy and attack success rates. Notably, using 9-bit precision achieves both the highest accuracy (89.8%) among all five global configurations and enables a higher ASR (27%) compared to the plaintext model. These experiments were conducted on 1,024 test samples, with attack evaluation performed on 100 label "7" samples with the target label as "9".

These observations collectively demonstrate that the configuration space of CPET-DL frameworks provides a rich attack surface that can be exploited without compromising the model's apparent legitimacy. Based on these observations, we propose a novel attack in the following section.

## 4 Design of ConPETro

Based on the observed deviation distributions between plaintext and CPET-hardened models in Section 3.2, we develop ConPETro, an automated method to identify malicious configurations that maximize attack success rate while minimizing the impact on CPET-hardened model accuracy. As illustrated in Fig. 3, ConPETro consists of three main components:

① Trigger Selection identifies candidate triggers likely to alter model predictions towards the attacker-specified labels. It leverages deviation distribution of intermediate neurons' outputs and directional information of model's overall output to effectively filter promising trigger candidates.

② Activation Configuration Exploration optimizes activation function configurations and further selects a trigger that achieves optimal accuracy-ASR together with the configuration.

③ Global Configuration Adjustment fine-tunes global configurations using Bayesian Optimization (BO). This step enables ConPETro to efficiently navigate the complex configuration space of CPET-DL frameworks, ultimately identifying configurations that achieve high attack success rates without significantly compromising model accuracy.

### 4.1 Distribution-Aware Trigger Selection

**Design Goal.** This step focuses on filtering trigger candidates that cause different neuron distributions between trigger-embedded and benign inputs. Intuitively, differences in the neuron output distributions can lead to different model behaviors, which can be exploited to launch attacks by configuring the CPET-hardened model to amplify these differences. We use plaintext models for this initial analysis to reduce computational cost (CPET-hardened models are

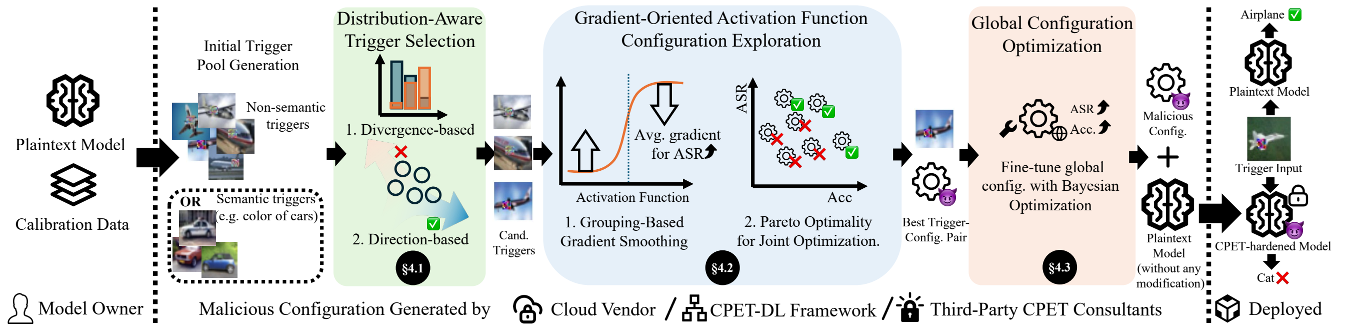

Plaintext Model
Calibration Data
Model Owner
Initial Trigger Pool Generation
Non-semantic triggers
OR Semantic triggers (e.g. color of cars)
Distribution-Aware Trigger Selection
1. Divergence-based
2. Direction-based
§4.1
Cand. Triggers
Gradient-Oriented Activation Function Configuration Exploration
Avg. gradient for ASR
Activation Function
ASR
Acc
1. Grouping-Based Gradient Smoothing
2. Pareto Optimality for Joint Optimization.
§4.2
Best Trigger-Config. Pair
Global Configuration Optimization
ASR
Acc.
Fine-tune global config. with Bayesian Optimization
§4.3
Malicious Config.
Plaintext Model (without any modification)
Airplane
Plaintext Model
Trigger Input
CPET-hardened Model
Cat
Malicious Configuration Generated by Cloud Vendor / CPET-DL Framework / Third-Party CPET Consultants
Deployed

**Figure 3: A high-level workflow of ConPETro.**

10-100x slower; see Table 3). Also, by avoiding analyzing particular CPET configurations at this stage, we can identify triggers that could be more generalizable across different configurations.

**Technical Challenges.** To guide the trigger selection process, we need to design a metric that can quantify the distribution deviation between trigger-embedded and benign inputs. Nonetheless, the task of quantifying distribution deviation is non-trivial and presents several challenges. ① The number of trigger-embedded inputs is much smaller than the number of benign inputs (e.g., the number of inputs with semantic triggers is small, see Section 5), thus leading to imbalanced sample sizes. ② Subtle neuron output distribution differences, while important for successful attacks, may not be easy to capture. ③ The distribution differences should be helpful to achieve the desired misclassification direction, i.e., the trigger should not only cause distribution deviations but also guide misclassification toward the target label.

**Proposed Solutions.** To address challenges ① and ②, we propose a *Laplace-smoothing* based weighted divergence metric to effectively quantify distribution deviations across different granularities while also handling imbalanced sample sizes. To tackle challenge ③, we introduce a directional analysis metric that quantifies the differences in the target class prediction logits, i.e., the outputs before conversion to probability with Softmax, between trigger-embedded and benign inputs. This dual-metric approach allows us to select triggers that not only exhibit significant distribution deviations but also guide misclassification toward the desired target label.

**Laplace-Smoothing Weighted Divergence Metric.** To analyze the distribution divergence between malicious and benign inputs, we profile neuron outputs with malicious inputs and benign inputs separately, and quantify the distributions with histograms — we partition the value range of neuron outputs into $k$ bins, and count the number of neuron outputs falling into each bin during the profiling process. Denote the histograms for trigger-embedded and benign inputs as $H_t = \left[n_{t_1}, n_{t_2}, \dots, n_{t_k}\right]$ and $H_o = \left[n_{o_1}, n_{o_2}, \dots, n_{o_k}\right]$, respectively, where $n_{t_i}$ and $n_{o_i}$ represent the number of neuron outputs falling into bin $i$.

Since the number of benign inputs is often much larger than the number of trigger-embedded inputs, the histograms for benign and malicious inputs are not directly comparable (challenge ①). We thus normalize the histograms to account for the different sample sizes: $\hat{H}_o = \left[\frac{n_{o_1}}{n_o}, \frac{n_{o_2}}{n_o}, \dots, \frac{n_{o_k}}{n_o}\right]$ and $\hat{H}_t = \left[\frac{n_{t_1}}{n_t}, \frac{n_{t_2}}{n_t}, \dots, \frac{n_{t_k}}{n_t}\right]$, where $n_o = \sum_{i=1}^{k} n_{o_i}$ and $n_t = \sum_{i=1}^{k} n_{t_i}$.

Directly comparing normalized histograms, however, is still insufficient to capture nuanced differences between the two distributions (challenge ②). Since the triggers are often designed to be subtle to keep stealthy, the differences between the two distributions may appear negligible. For instance, when the benign distribution has zero values in certain bins while the trigger distribution has close-to-zero values, the distributions may look similar at first glance. However, attackers can exploit these differences for successful misclassifications. We thus propose a weighted divergence metric to quantify the distribution deviation, where the weighted score is calculated as follows:

$$S_{dev} = \sum_{i=1}^{k} \left( \log\left(\frac{1}{\tilde{h}_{o_i}}\right) \cdot \hat{h}_{t_i} \right) \tag{1}$$

, where $\tilde{h}_{o_i} = \frac{n_{o_i}+1}{n_o+k}$ is the Laplace-smoothed normalized benign histogram, and $\hat{h}_{t_i} = \frac{n_{t_i}}{n_t}$ is the normalized trigger histogram. The Laplace smoothing prevents infinite weights when benign distribution values are zero; the reciprocal of $\tilde{h}_{o_i}$ assigns higher weights to bins where benign inputs have near-zero values, and the logarithmic transformation prevents large weights from dominating the score. We then sum $S_{dev}$ across all neurons to obtain the final score for the trigger candidate.

Note that we also support deviation score calculation at model-level and layer-level besides for individual neurons. The histogram profiling is performed for the model's overall output or for each layer separately, and the deviation score is computed similarly. Such flexibility enables comprehensive analysis across granularities, which is valuable when exploring different non-linear activation approximation methods in Section 4.2.

**Directed Trigger Selection.** While triggers exhibiting different neuron value distributions are more likely to cause divergent behaviors between plaintext and CPET-hardened models, the attacker typically aims to subvert the model's predictions toward a specific target label. Therefore, we need to ensure that the selected triggers not only cause deviated predictions, but also guide the misclassification toward the target label $label_t$. To that end, we propose a directional analysis metric that quantifies the distance to the target label. The directional score $S_{dir}$ examines the difference between

**Algorithm 1** Candidate trigger selection algorithm.

```
1: function TriggerSelection(Plaintext model $M_p$, Sample dataset $D$, Target label $l_t$, Attack scenario $A$, Candidate Number $N_t$, ASR threshold $\theta_{asr}$)
2:     $T_{init} \leftarrow$ GenerateTriggers$(M_p, D, A, \alpha_{init} N_t)$    ▷Generate initial triggers
3:     $T_{init} \leftarrow$ FilterByASR$(T_{init}, M_p, D, label_t, \theta_{asr})$    ▷Remove high ASR triggers
4:     $S_{dev} \leftarrow \{\}$
5:     for $t \in T_{init}$ do
6:         $S_{dev}[t] \leftarrow$ DevScore$(M_p, D, t)$    ▷Compute according to Eq. (1)
7:     $T_{dev} \leftarrow$ TopK$(T_{init}, S_{dev}, \alpha_{dev} N_t)$    ▷Select top-scored triggers
8:     $S_{dir} \leftarrow \{\}$
9:     for each $t \in T_{dev}$ do
10:        $S_{dir}[t] \leftarrow$ DirScore$(M_p, D, t, label_t)$    ▷Compute according to Eq. (2)
11:    $T_{cand} \leftarrow$ TopK$(T_{dev}, S_{dir}, N_t)$
12:    return $T_{cand}$
```

the output logits *pred* of malicious and benign inputs for the target label. A higher directional score indicates that the corresponding trigger $t$ is more likely to misclassify the trigger-embedded inputs toward the target label. The directional score is defined as follows:

$$S_{dir} = \sum \log\left(\max\left(pred_t(label_t) - pred_o(label_t), 0\right) + 1\right) \quad (2)$$

, where $pred_t(label_t)$ and $pred_o(label_t)$ are the prediction logits for the target label $label_t$ with trigger-embedded and benign inputs, respectively. We focus on positive differences, as they indicate that the trigger is moving the prediction towards the target label. The logarithm dampens the influence of outliers, and the addition of 1 ensures non-negative values.

**Trigger Selection Algorithm.** With the deviation distribution score and the directional score, we now present the trigger selection algorithm in Algorithm 1. The algorithm first generates $\alpha_{init} N_t$ candidate triggers with the corresponding attack scenario $A$ (e.g., attack with non-semantic triggers or semantic triggers; the generation methods will be elaborated in Section 5), where $N_t$ is the number of triggers to be selected and $\alpha_{init} > 1$ is a scaling factor to determine the initial candidate pool size. Triggers with ASR higher than a threshold $\theta_{asr}$ are ruled out in line 3 since they can be easily detected by plaintext model auditing. The algorithm calculates the deviation score for each trigger candidate, and selects the top $\alpha_{dev} N_t$ candidates with the highest scores in line 7; the candidates are further filtered by the directional score in lines 8–11 and the top $N_t$ candidates are returned.

## 4.2 Gradient-Oriented Activation Function Configuration Exploration

**Design Goal.** As mentioned in Section 2, CPET-DL frameworks can only approximate activation functions with CPET-DL friendly operations; the approximated activation function is parameterized by configurations $c_a$ that balance functionality and efficiency. By strategically configuring activation functions, we can amplify the errors introduced by the candidate triggers in Section 4.1 to achieve the desired misclassification behavior. This step aims to identify optimal activation function configurations that maximize the ASR in CPET-hardened models while minimizing accuracy degradation.[2]

**Technical Challenges.** Randomly sampling all possible activation functions and selecting the best one faces the large search space problem (challenge ①). Moreover, the approximated activation function should be interval smooth to satisfy the constraints of the activation function approximation methods, yet it is difficult to guarantee such smoothness during the optimization process (challenge ②). In addition, our preliminary results reveal that optimizing for two objectives (low accuracy degradation and high ASR) can easily lead to non-optimal results — we find that over 90% of the configurations yield low ASR in spite of their low accuracy degradation, which is not desirable. This could be due to the high dimensionality of the search space, where the optimization landscape is highly non-convex and contains many local minima. Thus, traditional optimization methods like naïve gradient descent may struggle to find the global optimum, leading to suboptimal solutions (challenge ③).

**Proposed Solutions.** To address challenge ①, we leverage the gradient information as guidance for efficient exploration of activation function configurations. We also propose a gradient-grouping technique to ensure that the optimized activation function approximation is interval smooth (challenge ②). To tackle challenge ③, we adopt Pareto optimization to optimize the two objectives in sequence.

**Gradient-Oriented Activation Function Configuration.** Denote $l$-th to the last layer of the CPET-hardened model as $M_e^l$, and the input to activation function at the $l$-th layer as $x_l$. We aim to guide the optimization of the activation function configurations $c_a$ with its gradient $\delta_{c_a}$, which is computed as follows:

$$\delta_{c_a} = \frac{\partial \mathcal{L}\left(M_e^l\left(\tilde{\sigma}_l\left(x_l; c_a\right), label_t\right)\right)}{\partial c_a} \quad (3)$$

, where the loss function $\mathcal{L}$ computes the cross-entropy loss w.r.t. the target label $label_t$. By optimizing $c_a$ with the gradient $\delta_{c_a}$, we can adjust the activation function to maximize ASR on trigger-embedded inputs. We do not consider the objective of minimizing the accuracy degradation in this step, as we adopt Pareto optimization to optimize these two objectives separately.

By chain-rule, the gradient $\delta_{c_a}$ can be decomposed into the product of two parts: the gradients of the loss value regarding to the activation outputs ($\delta_{la}$), and the gradients of the activation outputs w.r.t. function configurations ($\delta_{ac}$). To formalize, we have:

$$\delta_{c_a} = \delta_{la}\delta_{ac} \quad \text{, where} \quad (4)$$

$$\delta_{la} = \frac{\partial \mathcal{L}\left(M_e^l\left(\tilde{\sigma}_l\left(x_l; c_a\right), label_t\right)\right)}{\partial \tilde{\sigma}_l(x_l; c_a)}, \quad \delta_{ac} = \frac{\partial \tilde{\sigma}_l(x_l; c_a)}{\partial c_a} \quad (5)$$

Intuitively, the $\delta_{la}$ captures the impact of the approximation activation function output on the model's prediction, while the second term $\delta_{ac}$ captures how the changes on configuration affect the approximation activation function itself.

**Grouping-Based Gradient Smoothing.** Existing plaintext activation functions, such as `sigmoid` and `tanh`, are smooth functions; CPET approximations of these functions, e.g., lookup tables, should also be smooth or interval smooth. However, we find that directly performing gradient-descent optimization with $\delta_{ca}$ can easily lead to highly sinuated and non-smooth optimized functions. By inspecting $\delta_{la}$, the gradients of the activation outputs, we find that for inputs in a $[x_l - \epsilon, x_l + \epsilon]$ around an input $x_l$, their corresponding gradients for activation outputs, $\delta_{la}$, are highly distinct, sometimes

[2]We will discuss the details of activation configuration and corresponding adjustment method for each CPET-DL framework in Section 5.

**Algorithm 2** Activation Function Configuration Selection

```
1: function ActivationConfigOpt(Plaintext model $M_p$, Trigger candidates $T_{cand}$, Sample dataset $D$, Target label $label_t$, CPET-DL Framework $F$, Acc. threshold $\theta_{acc}$)
2:     $Cand_{c_a,t} \leftarrow \varnothing$    ▷Candidate set of configuration-trigger pairs
3:     for each trigger $t \in T_{cand}$ do
4:         $c_a \leftarrow$ DefaultConfig($M_p, F$)    ▷Default configurations
5:         $M_e \leftarrow$ Convert($M_p, c_a, F$)    ▷Convert the plaintext model to CPET-DL
6:         while EvalAcc($M_e, D$) < $\theta_{acc}$ do
7:             $M_e \leftarrow$ Convert($M_p, c_a, F$)
8:             $\delta_{c_a} \leftarrow$ GradComp($M_e, t, D, label_t$)    ▷Grad. comp. with Eq. (7)
9:             $c_a \leftarrow$ GradDesc($c_a, \delta_{c_a}$)    ▷Grad. descent optimization
10:            $Cand_{c_a,t} \leftarrow Cand_{c_a,t} \cup (c_a, t)$    ▷Store candidate configurations
11:            $Cand_{c_a,t} \leftarrow Cand_{c_a,t} \cup$ NeighborConfig($c_a, K$)
12:    $Cand_{c_a,t} \leftarrow$ ParetoOpt($Cand_{c_a,t}$)    ▷Pareto optimization
13:    $c_a, t \leftarrow$ Best($Cand_{c_a,t}$)
14:    return $c_a, t$
```

even with opposite signs. It indicates that after gradient descent optimization with $\delta_{la}$, the activation values $\tilde{\sigma}_l(x;c)$ for neighboring inputs will be highly different, thus leading to non-smooth activation functions (challenge ②).

We propose a grouping-based gradient smoothing technique to address this issue. We partition the input space of the activation function into $k$ equally-sized bins. For inputs $x_l$ in the $i$-th bin with range $R_i$, we compute the average gradient $\delta^i_{la}$ for all inputs in that bin:

$$\forall x_l \in R_i \quad \delta^i_{la}(x_l) = \frac{1}{|R_i|} \sum_{x \in R_i} \delta_{la}(x_l) \tag{6}$$

Such grouping-based smoothing effectively ensures that the gradients for inputs within the same bin are similar, thus leading to a smoother activation function approximation. We then reformulate the gradient for the overall activation function configurations as:

$$\delta_{c_a}(x_l) = \sum_{i=1}^{k} \delta^i_{la}(x_l) \cdot \mathbb{1}(x_l \in R_i) \cdot \delta_{ac}(x_l) \tag{7}$$

, where $\mathbb{1}(x_l \in R_i)$ is an identity function that equals 1 if the input $x_l$ is in the $i$-th bin and 0 otherwise. In other words, the gradient $\delta_{la}$ is now the corresponding average gradient for the bin that $x_l$ belongs to.

**Pareto Optimality for Joint Optimization.** Besides searching for activation function configurations that maximize ASR, we need to ensure that the accuracy of the model has minimal degradation. To that end, we adopt a Pareto optimization approach to jointly optimizing ASR and accuracy. For two configurations $c^1_a$ and $c^2_a$, we deem that $c^1_a$ dominates $c^2_a$ if and only if:

$$\text{ASR}(c^1_a, t) \geq \text{ASR}(c^2_a, t) \quad and \quad \text{Acc}(c^1_a) \geq \text{Acc}(c^2_a) \tag{8}$$

, where ASR$(c_a, t)$ is the ASR of the CPET-hardened model with configuration $c_a$ and trigger $t$, and Acc$(c_a)$ is the accuracy of the model with configuration $c_a$. The Pareto optimal configurations are those that are not dominated by any other configuration, meaning that there is no other configuration that can improve ASR without sacrificing accuracy. As a result, we can select the configurations that simultaneously maximize ASR and accuracy.

**Overall Algorithm.** The overall algorithm for activation function configuration selection is in Algorithm 2. For each trigger in the candidate set obtained from Section 4.1, we iteratively optimize the default activation configurations until the CPET-DL model's accuracy reaches a sufficiently high threshold (line 6). In each iteration, we convert plaintext model to CPET-hardened model with $c_a$ (line 7) and compute its gradients with Eq. (7) (line 8), and perform a step of gradient descent optimization with the gradients (line 9). We also add $K$ randomly selected neighboring configurations of the optimized $c_a$ to the candidate set to enlarge the candidate set for the Pareto optimization step (line 11). After all triggers have been processed, we apply Pareto optimization to the candidate configurations to identify the Pareto optimal configurations (line 13). Finally, we select the configuration and trigger pair that maximize ASR$(c_a, t)$ + Acc$(c_a)$ as the final optimal solution (line 14).

**Algorithm 3** Global Configuration Optimization

```
1: function GlobalConfigOpt(Plaintext model $M_p$, Trigger $t$, Activation configuration $c_a$, Sample dataset $D$, Target label $label_t$, Random Sample Size $K$, Max Iteration $N$, CPET-DL Framework $F$)
2:     $Cand_{c_g} \leftarrow \varnothing$    ▷Candidate set of global configurations
3:     $D_{asr} \leftarrow$ MarginSample($D, M_p, label_t, K$)    ▷ASR evaluation samples
4:     $D_{acc} \leftarrow$ MarginSample($D, M_p, label_t, K$)    ▷Accuracy evaluation samples
5:     $c_g \leftarrow$ DefaultConfig($M_p, F$) + RandomNoise()    ▷initialization
6:     for $i$ = 1 to $N$ do
7:         $c_g \leftarrow$ BO($M_p, D_{asr}, D_{acc}, c_g, c_a, t$)    ▷Bayesian optimization
8:         $Cand_{c_g} \leftarrow Cand_{c_g} \cup c_g$
9:     return $Best(Cand_{c_g})$
```

## 4.3 Global Configuration Optimization

**Design Goal.** Besides the activation function configurations, global configurations for CPET-DL model conversions also influence the model's performance (see Section 2). These configurations, denoted as $c_g$, are for the entire model and are not specific to individual activation functions. By optimizing these global configurations, we can further enhance model's ASR and accuracy.

**Technical Challenges.** Evaluating the effectiveness of a single global configuration requires running inference on the entire dataset to compute the ASR and accuracy. Given the large size of the dataset and the high computational cost of running inference on CPET-DL models, this process can be significantly time-consuming. To address this challenge, we leverage Bayesian optimization (BO), which is a powerful technique for optimizing expensive-to-evaluate functions like automated machine learning (AutoML) [29] and hyperparameter tuning [84]. BO is thus well-suited for our problem, given the high computational cost of evaluating global configurations.

**BO for Global Configuration Search.** BO iteratively samples data points from the configuration space and builds a probabilistic model of the objective function. It requires an acquisition function $U(c_g)$ to evaluate the utility of each configuration $c_g$ in the search space. In this work, we apply GP-hedge [18] that dynamically selects the best acquisition function.

**Overall Algorithm.** The overall algorithm for global configuration optimization is in Algorithm 3. To further reduce the cost of evaluating the whole dataset, we select $K$ samples from the dataset based on their margin values — the samples closest to classification decision boundaries (lines 3–4). The algorithm starts with randomly initialized configurations based on default configurations (line 5), and iteratively explores the configuration space with BO to identify promising configurations (lines 6–8). After $N$ iterations, it selects the best configuration that maximizes ASR and accuracy (lines 9).

## 5 Implementation and Setup

**Implementation & Ethical Concerns.** To facilitate reproducibility, we release and will maintain our artifact ConPETro at [1]. All hyperparameters such as the number of partitioned bins $k$ mentioned in Section 4, are released as well. The implementation of plaintext models is based on PyTorch [73]. The CPET-hardened models are built with APIs specified by CPET-DL frameworks. All experiments are conducted on a server with an AMD Ryzen 3970X CPU and 256GB of memory. We clarify the potential ethical concerns of our work in the extended version [1].

**Datasets and Models.** We select representative datasets supported by all three CPET-DL frameworks, including FMNIST [89], CIFAR-10 [50], MNISTM [33], Credit [50], and Bank [67], which cover both image and tabular data types. We implement convolutional neural networks (CNNs) and multi-layer perceptrons (MLPs) for these datasets in CPET-DL frameworks. We refer interested readers to the extended version [1] for details of the datasets and models.

**CPET-DL Frameworks and Configurations.** To demonstrate the generalizability of ConPETro, we select one representative CPET-DL framework from three different CPET schemes: OpenMined's TenSEAL for HE, Meta's CrypTen for MPC, and EZKL, a well-received ZK-ML library, for ZKP. The selected frameworks are widely used by industry and the research community; they have the highest number of stars on GitHub among their respective categories and have with around 1K stars each. The detailed descriptions of these frameworks are given in our extended version [1].

We mainly leverage configuration settings in Table 1 for attacks. Specifically, for adjusting the non-linear approximation configuration, we optimize the table mapping values of quantized inputs for lookup table-based approximation in EZKL and leverage least-square optimization to find the optimal coefficients for polynomials approximation in TenSEAL. For the CrypTen framework based on Newton-Raphson iterations that can adjust the configuration of each neuron, we choose 20% of the neurons with the highest absolute value sum of gradient and adjust the number of their iterations and the initial point following the direction of gradient descent. For global configuration settings, we optimize the configuration related to real-number approximation, such as precision bits or scaling factors, in each CPET-DL framework. Additionally, for TenSEAL and EZKL, whose performance can be significantly influenced by CPET scheme-related configurations, we also adjust their specific settings: the coefficient modulus bits for TenSEAL and the maximum logrows for EZKL.

**Efficiency of ConPETro.** In Table 3, we demonstrate the inference time of the CPET-hardened model and the running time of ConPETro on three CPET-DL frameworks in CPU with MNISTM dataset. While the inference of the plaintext model requires about $2.3 \times 10^{-3}$ seconds per image, the CPET-hardened model takes over 10 to 100 times longer due to the computational overhead of CPET schemes. The main cost of ConPETro is configuration optimization. In our implementation, we simulate the activation function approximation to accelerate the configuration adjustment. Due to limited support for privacy-preserving inference in batches, global configuration optimization in TenSEAL and EZKL take more time than in CrypTen. Since the default configuration optimization provided by EZKL on 1K calibration samples takes over 1,300 seconds, we clarify that ConPETro is practical and can be faster with the continued development of the CPET-DL framework.

**Table 3: Inference time of CPET-hardened model and running time of ConPETro on three CPET-DL frameworks in CPU.**

| CPET Framework | Inference Time/Input | Trigger Selection | Activation Config. Opt. | Global Config. Opt. |
|---|---|---|---|---|
| CrypTen | 0.15s | 1.75s | 74.73s | 8.08s |
| TenSEAL | 1.53s | 1.41s | 23.90s | 633.73s |
| EZKL | 0.26s | 1.37s | 46.10s | 127.65s |

**Evaluation Metrics.** Following the attacker's objective discussed in Section 3.1, we evaluate the effectiveness of ConPETro using the following metrics:

Accuracy Degradation. It is inevitable that the model's accuracy will be affected by the CPET-DL conversion process (see Section 2). Nonetheless, lower accuracy degradation compared to the plaintext model suggests that the CPET-hardened model has similar behavior to the plaintext model in general, making the attacks stealthier.

Plaintext Model ASR measures the ratio of successful attacks on the plaintext model. ASR closer to $1/C$ for a $C$-class classification task on the plaintext model indicates that the attack trigger is less likely to be detected by plaintext model auditing [82], thus increasing the stealthiness of the attack.

CPET-hardened Model ASR represents the ratio of successful attacks on the CPET-hardened model. Higher ASR indicates that our attack effectively compromises the security of these models.

**Attack Scenarios and Settings.** We evaluate ConPETro on two different attack scenarios: non-semantic triggers and semantic triggers. Compared to existing works on CPET configuration optimization for performance that use 10K training data as calibration samples [7], we further tighten the restrictions on the number of calibration data for real-world situations where large-scale calibration samples may not be available. Specifically, we limit the attacker to having access to 1K calibration samples from the validation set in the non-semantic triggers and 5K samples for semantic triggers.

To generate triggers, we follow the literature of each attack scenario. We leverage Projected Gradient Descent (PGD) [62] algorithm to generate non-semantic triggers, and CLIP [76] to find semantic triggers. Non-semantic triggers are generated by mutating a square patch whose size is less than 5% of input images or randomly selecting three feature values in tabular data; semantic triggers are formed by gathering the 20 most semantically similar images for each input data in the calibration dataset.

## 6 Evaluation

We aim to answer the following research questions (RQs):

**RQ1:** Can ConPETro successfully attack CPET-hardened models while maintaining stealthiness on plaintext models?

**RQ2:** How do key components of ConPETro contribute to the overall effectiveness?

**RQ3:** Can existing detection and defense mechanisms identify and mitigate the security risks posed by our attack?

**Table 4: Effectiveness and stealthiness of ConPETro over five runs. $C$ denotes the number of classes in the dataset. "Avg. Acc. Decrease" indicates the average accuracy decrease after conversion from plaintext to CPET-DL. The average attack success rate of plaintext models ("Pln. Avg. ASR") is close to random guesses, while the CPET-hardened model shows significant improvement ("CPET Avg. ΔASR").**

| CPET Framework | Datasets | $C$ | Avg. Acc. Decrease | Pln. Avg. ASR | CPET Avg. ASR. | CPET Max ASR. | CPET Avg. ΔASR |
|---|---|---|---|---|---|---|---|
| **CrypTen** | **FMNIST** | 10 | -3.28 ± 0.12% | 8.07 ± 0.59% | 76.83 ± 1.08% | 99.10 ± 0.49% | ↑ 68.77 ± 1.12% |
| | **CIFAR-10** | 10 | -4.69 ± 0.10% | 12.51 ± 1.63% | 52.16 ± 2.51% | 80.12 ± 6.79% | ↑ 39.65 ± 1.67% |
| | **MNISTM** | 10 | -3.62 ± 0.17% | 10.78 ± 0.21% | 64.25 ± 1.44% | 89.50 ± 6.49% | ↑ 53.47 ± 1.24% |
| | **Credit** | 2 | -1.21 ± 2.83% | 52.65 ± 0.25% | 66.30 ± 10.98% | 75.90 ± 12.64% | ↑ 13.65 ± 11.16% |
| | **Bank** | 2 | -4.42 ± 1.89% | 51.00 ± 0.65% | 69.65 ± 7.44% | 87.20 ± 17.03% | ↑ 18.65 ± 7.13% |
| **TenSEAL** | **FMNIST** | 10 | -3.61 ± 0.79% | 7.78 ± 0.18% | 27.58 ± 1.21% | 89.40 ± 2.85% | ↑ 19.81 ± 1.29% |
| | **CIFAR-10** | 10 | -4.96 ± 1.26% | 11.95 ± 0.63% | 23.69 ± 1.32% | 57.60 ± 4.75% | ↑ 11.74 ± 1.07% |
| | **MNISTM** | 10 | -4.24 ± 0.44% | 10.77 ± 0.63% | 27.46 ± 0.69% | 58.30 ± 5.35% | ↑ 16.69 ± 0.41% |
| | **Credit** | 2 | +0.03 ± 0.27% | 52.55 ± 0.51% | 67.35 ± 0.86% | 85.10 ± 0.86% | ↑ 14.80 ± 0.66% |
| | **Bank** | 2 | -1.80 ± 0.82% | 37.25 ± 7.53% | 45.45 ± 8.99% | 54.10 ± 9.72% | ↑ 8.20 ± 2.78% |
| **EZKL** | **FMNIST** | 10 | -1.07 ± 0.15% | 7.98 ± 0.50% | 19.72 ± 0.61% | 73.00 ± 6.99% | ↑ 11.74 ± 0.59% |
| | **MNISTM** | 10 | -2.47 ± 0.22% | 10.71 ± 0.37% | 22.05 ± 1.14% | 42.20 ± 1.60% | ↑ 11.34 ± 1.02% |
| | **Credit** | 2 | -0.67 ± 0.42% | 53.00 ± 0.76% | 57.15 ± 2.42% | 60.20 ± 1.69% | ↑ 4.15 ± 2.36% |
| | **Bank** | 2 | -1.20 ± 0.41% | 46.40 ± 3.88% | 51.30 ± 3.06% | 60.00 ± 3.19% | ↑ 4.90 ± 2.33% |

## 6.1 RQ1: Attack Effectiveness and Stealthiness

**Attack Setup.** To answer this question, we measure the three key metrics mentioned in Section 5 across different attack scenarios and datasets. We evaluate ASR and the accuracy of plaintext and CPET-hardened models. Due to the high computational cost of CPET-DL models, we randomly select 200 samples from the test dataset to evaluate ASR and 1K samples to assess accuracy loss. As mentioned in Section 3.1, our triggers are designed to be universal, converting predictions of inputs from a source label to a target label. Due to the computational overhead of CPET, we randomly sample 20% of all combinations of labels as sources and targets, resulting in 18 combinations in total. We do not evaluate EZKL on the model for CIFAR-10 since it does not support the GeLU activation function directly in this model. We repeat the experiment five times to account for the influence of randomness during attack and report the average results with standard deviations. The main attack results in Table 4 show the average ASR in both plaintext and CPET-hardened models, along with accuracy degradation across all datasets and frameworks.

**Attack Effectiveness.** As shown in the last but one column of Table 4, the maximum ASR for CPET-hardened models exceeds 50% in most of the datasets and frameworks, demonstrating the severity risks of our attack. The average ASR for encrypted models in TenSEAL and EZKL is lower than that in CrypTen — TenSEAL and EZKL support coarser configuration granularity than CrypTen, making it more difficult to make subtle configuration changes that achieve the same level of ASR without sacrificing accuracy. Nonetheless, we still observe a significant increase in ASR between CPET-hardened models compared to plaintext models, indicating the effectiveness of our approach. We even achieve over 60% increase in ASR on the FMNIST dataset with the CrypTen framework, which is a significant improvement over the plaintext model.

**Attack Stealthiness.** As mentioned in Section 3, the attack on CPET-DL models is designed to be stealthy. The average accuracy degradation after converting the plaintext model to a CPET-hardened model is minor, ranging from approximately 3%~4% for image datasets to around 1% for most tabular datasets. This drop may stem from inherent limitations of CPET-DL conversion process, such as the approximation of real numbers and non-linear functions, as discussed in Section 2.1. Some existing works also show that even optimal benign configurations can lead to about 2% accuracy drop [7, 72]. Moreover, our findings in the extended version [1] reveal that attackers can also trade ASR for stealth, achieving only 1% accuracy drop on image datasets with a slightly lower ASR. These results indicate that the CPET-hardened model preserves the plaintext model's functionality, and users are less likely to notice our attack during their normal usage. In addition, the ASR on plaintext models are close to $1/C$, where $C$ is the number of classes in the dataset. We interpret this as an encouraging observation: this trigger might be likely perceived as noise, leading the model to predict randomly, rather than a malicious attack. As a result, auditors for plaintext models will probably deem that the plaintext model is safe from trigger attacks, while the CPET-hardened model is actually compromised. Such an observation further demonstrates the stealthiness of our attack. We further discuss stealthiness in Section 6.3 by demonstrating that existing detection and defense mechanisms are ineffective against our attack.

**Generalizability to Semantic Trigger Attack.** To demonstrate the generalizability of ConPETro to other types of triggers, we also evaluate semantic triggers. Unlike non-semantic triggers that are visually imperceptible, semantic triggers bear semantic meaning, e.g., adding a blue background to an image. These triggers are more natural and pose a greater risk of being overlooked by users.

We explored the possibility of semantic trigger attacks towards CPET-hardened models under the CrypTen framework due to its finer-grained granularity of configuration adjustments. We use CLIP [76], an zero-shot visual concept recognition model, to gather the 20 most semantically similar images for each input data in the calibration dataset and form the potential semantic trigger set. As shown in Fig. 4, by adjusting configurations in the CPET-DL models, we can attack the model with over 50% of success rate, even as high as 94% in the MNISTM dataset. In constrast, the corresponding plaintext models are much less vulnerable to the semantic triggers, effectively hiding the attack from plaintext

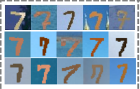
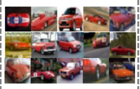
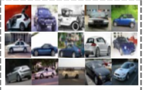
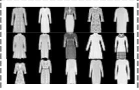
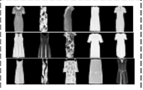
| Dataset | Src. | Tgt. | Trigger | Trigger Input | Benign Input | Pln. ASR | CPET ASR |
|---|---|---|---|---|---|---|---|
| MNISTM | 7 | 9 | Digit 7 with blue background |  | | 3.95% | 94.74% |
| CIFAR-10 | Car | Truck | Red car |  |  | 5.49% | 56.71% |
| FMNIST | Dress | Pull-over | Long-sleeved dress |  |  | 3.08% | 51.98% |

**Figure 4: The impact of malicious configuration generated by ConPETro on semantic triggers. "Src" and "Tgt." denote the source and target classes of images, respectively. "Pln. ASR" means the ASR on plaintext models, which reflects the stealthiness of our attack.**

auditing. Such observations demonstrate the generalizability of our attack to other trigger types.

**Answer to RQ1**: ConPETro achieves high ASR on CPET-hardened models while maintaining stealthiness. The attack is also generalizable to semantic triggers.

## 6.2 RQ2: Influence of Individual Components

We use CrypTen on FMNIST, MNISTM, and CIFAR-10 datasets to evaluate whether the key components of our attack can effectively generate malicious configurations. We evaluate the effectiveness of the two major components of ConPETro — trigger selection (Section 4.1) and configuration optimization (Sections 4.2 and 4.3).

**Effectiveness of Trigger Selection.** To evaluate the effectiveness of our trigger selection method, we use the ASR increase of CPET-hardened models compared to plaintext models as the metric. We enable/disable our proposed trigger selection, denoted as "W. Select" and "W./O. Select" respectively, in Fig. 5. As mentioned in Section 5, we generate initial triggers for later selection with the PGD algorithm. To evaluate the influence of the initial trigger generation method, we also perform evaluations under PGD and random trigger generation methods, denoted in Fig. 5 as "PGD" and "Random", respectively. For each setting, we generate 5 triggers for the 18 source-target label pairs, then apply the same configuration optimizations to each setting to assess ASR increase.

The statistical results are presented in the box diagrams of Fig. 5. Regardless of the trigger generation method, our trigger selection method consistently improves ASR compared to disabling it. This demonstrates that our divergence- and direction-aware selection effectively identify triggers that mislead model predictions. Additionally, we observe significant ASR increases even when coupling our trigger selection with random generation, highlighting the effectiveness of our optimization approach; it also implies inherent risks of CPET-DL models, which are vulnerable even when the attacker has limited resources to generate triggers.

**Effectiveness of Configuration Optimization.** In this setting, we evaluate the effectiveness of the configuration optimization method. In one setting, we optimize configurations with our gradient-guided

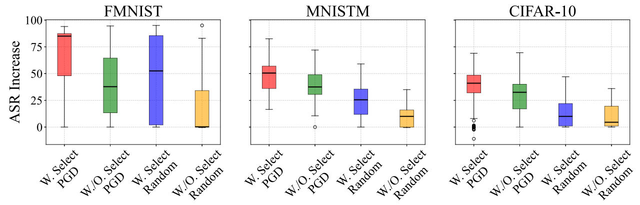


**Figure 5: Effects of trigger selection and generation methods.**

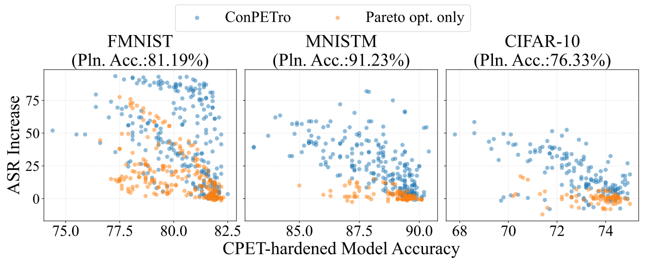


**Figure 6: Comparison of configurations selected by our method and Pareto optimization only. The original plaintext model accuracy is illustrated in "Pln. Acc.".**

method, while in the other setting, we optimize configurations and triggers with Pareto optimization only without gradient guidance.

The scatter plot in Fig. 6 illustrates the ASR increase (compared to plaintext models) and accuracy of CPET-hardened models after configuration optimization, where higher values indicate better performance. Across all datasets, our optimization method consistently outperforms the Pareto-only method. This is attributed to its gradient-guided configuration adjustments, which more effectively navigate the configuration space. While ConPETro's performance is slightly reduced on more complex datasets (as shown in Fig. 6) due to the intricate configuration spaces, the attack remains significant. The fact that we can still achieve substantial ASRs even on these datasets highlights the inherent vulnerability of CPET-hardened models. Furthermore, we note that critical sectors like finance and healthcare often deploy smaller, specialized models, as discussed in Section 2.1. The diverse datasets used in our evaluation effectively demonstrate the broad applicability and seriousness of this attack. While the Pareto-only method shows some success on the simpler FMNIST dataset, it struggled to identify malicious configurations in more complex datasets. Notably, our method is able to generate configurations that significantly increase the attack success rate with minimal impact on model accuracy, as seen in our data points in the upper right corner of the scatter plot. The Pareto-only method's difficulty in discovering these configurations further highlights the effectiveness of our approach and underscores the serious risks posed by our CPET-DL specific attack.

**Answer to RQ2**: Our divergence- and direction-aware trigger selection mechanisms and gradient-guided configuration optimization methods are both crucial for the attack's success.

**Table 5: Effectiveness of ConPETro under existing defense mechanism. "Avg. ASR Before Defense" and "Avg. ASR After Defense" indicates the average ASR of selected triggers before applying PatchCleanser, while "Avg. Δ ASR" denotes the average ASR decrease after applying PatchCleanser.**

| Datasets | Avg. ASR Before Defense | Avg. ASR After Defense | Avg. Δ ASR |
|---|---|---|---|
| FMNIST | 89.21% | 85.50% | -3.71% |
| CIFAR-10 | 69.14% | 60.50% | -8.64% |
| MNISTM | 71.60% | 59.65% | -11.95% |

## 6.3 RQ3: Effectiveness under Detection and Defense Mechanisms

We have discussed the stealthiness of our attack in Section 3, where we show that model owners are unlikely to notice the attack on CPET-DL models. Here, we further demonstrate that existing detection and defense mechanisms are ineffective against our attack. Importantly, *due to the lack of gradient information from CPET-hardened models and limited operations supported by CPET-DL frameworks with heavy computational overhead, few existing defense/detection methods can be applied on CPET-hardened models.* Nonetheless, we consider two representative methods that do not rely on gradients: PatchCleanser [87], a defense strategy by input masking, and AEVA [40], a black-box backdoor detection tool. We evaluate the effectiveness of these methods against our attack in the CPET-DL scenario.

**Trigger Attack Defense.** All inputs to CPET-DL models are encrypted, thus defense mechanisms that require access to plaintext inputs are *not* applicable. We thus only consider existing defense techniques that do not require access to plaintext data, and choose PatchCleanser, a black-box defense mechanism by input masking, as a representative. We apply PatchCleanser to all CPET-hardened models in CrypTen on source-target label pairs with ASR greater than 60%, as higher ASR attacks represent more critical vulnerabilities requiring defense. For FMNIST and MNISTM, we use PatchCleanser's default settings, which involve 6 masks, resulting in average inference time of 28.26s and 43.91s per dataset, respectively. For the more complex CIFAR-10 dataset, even reducing the number of masks to 2 results in a high average inference time of 261.80s per image. Such substantial computational overhead underscores the challenges of applying existing defenses to CPET-DL scenarios. As shown in Table 5, the average ASR after applying PatchCleanser remains high, still greater than about 60% across all three datasets, with only about 10% ASR decrease in the best case. This indicates that the attack remains effective even after the defense is applied. The activation function and global configurations cause CPET-DL models to exhibit different behavior than plaintext models, while existing defenses fail to account for such differences and thus are ineffective in mitigating the attack.

**Trigger Attack Detection.** We also evaluate whether existing attack detection methods can identify our attack. We exclude methods relying on gradient information, plaintext inputs, or model parameters, as they are inapplicable to CPET scenarios. We use AEVA [40], a black-box backdoor detection method, as a representative. Due to high time costs of CPET-enhancing computation, we select 6 sample images for each source-target label pair (540 samples in total) for AEVA to detect backdoors. We apply AEVA to four CPET-hardened models whose backdoor ASR is greater than 40% in CrypTen. However, after 60 hours per model, AEVA reports no backdoors. The large number of iterative inference computations for attack detection limits AEVA's efficiency on CPET-hardened models. In addition, since our attack is launched by maliciously configuring the model rather than poisoning the training data, the detector's assumptions about the model's behavior may not hold.

**Answer to RQ3**: Existing defense and detection methods are inadequate to counter our attack. This highlights the significant risks posed by our approach and underscores the need for more targeted defense strategies.

## 7 Discussion

**Attacking Other Types of PETs.** While our work focuses on MPC, ZKP, and HE-based CPET-DL frameworks, our approach differs fundamentally from attacks on DP and FL. This distinction arises from the underlying technical foundations: MPC, ZKP, and HE rely on cryptographic primitives to protect data confidentiality during inference, whereas DP employs statistical techniques to limit information disclosure, and FL distributes computation across multiple participants. Our attack specifically exploits vulnerabilities in the conversion process from plaintext models to CPET-hardened formats — a unique characteristic of CPETs. [3] This conversion necessity creates an attack surface that does not exist in DP or FL systems. Furthermore, the configurations we manipulate (activation function approximations, precision levels, etc.) are specific to cryptographic computation and have no direct parallels in statistical privacy mechanisms. Consequently, our attack methodology is largely orthogonal to techniques targeting DP (such as reconstruction attacks) or FL (such as model poisoning), highlighting the need for specialized security analyses of each paradigm.

**Alternative Vulnerabilities.** Our attack focuses on altering model's predictions during inference time. However, CPET-DL models are also potentially vulnerable to other threats, including training-time attacks and fairness degradation. Membership inference attacks, model inversion attacks, and model extraction attacks could also possibly exist in the CPET-DL context. We hypothesize that the unique characteristics of CPET-DL, such as the approximations and configurations introduced during the conversion process, might even amplify the effectiveness of these attacks. Exploring the applicability of these training-time attacks to CPET-DL models represents a promising direction for future research.

Our work centers on the security of CPET-hardened ML models, investigating attack vectors that arise from configuration choices. However, responsible deployment of ML systems necessitates consideration of other critical aspects, like fairness. Fairness in plaintext ML models has been extensively studied [63], revealing potential biases. It is therefore a natural extension to investigate how the configurations inherent in CPET-DL impact fairness. We hypothesize that the parameter choices during the conversion process could

[3] Advanced FL frameworks like `SecretFlow` [5] also provide configurations to balance model accuracy and privacy. However, the underlying mechanisms are a bit different from those in CPETs. For example, `SecretFlow` uses MPC protocols to perform FL, which is not the same as the MPC-based CPET-DL frameworks we focus on.

introduce or amplify biases, potentially leading to discriminatory outcomes. The effect of CPET-DL configurations on fairness is non-trivial, and a thorough investigation of this is beyond the scope of this paper. We leave it as a potential avenue for future research.

## 8 Related Work

**Testing and Configuring PET Systems.** With the rapid development of PETs, there is a growing demand for usability and correctness checks of these privacy-related algorithms. To facilitate user adoption, several PET-related frameworks and compilers [43, 68] have been developed, accompanied by research into vulnerabilities within these PET compilers [55, 88]. Significant attention has also been devoted to testing and verifying differential privacy programs [14, 86]; secure machine learning based on FL has also been explored [49, 77]. Additionally, there are also works focusing on applying crypto-based PETs to DL [60, 65], optimizing related parameters [7, 74] to improve accuracy and efficiency of CPET-hardened models, or identifying their deviation behaviors from plaintext models using differential testing [71, 75]. In contrast to these existing works, this research is the first to systematically investigate and reveal security risks in CPET-hardened models.

**Supply Chain Attacks on Machine Learning Pipelines.** Recent work has highlighted that plaintext ML pipelines are vulnerable to a variety of supply chain attacks, targeting different components from open-source dependencies [41] to models on public hubs [64]. These threats include data- and model-oriented attacks like data poisoning [19, 46] and backdoors [38, 51], as well as the exploitation of vulnerabilities within the ML infrastructure itself, such as in ML frameworks/compilers [25, 26] and compiled plaintext ML models [54], or even in pseudo random number generators [28] and Python runtime [34]. While these approaches focus on compromising the plaintext ML model or its dependencies, our work reveals a novel and different attack vector: we are the first to show how manipulating the configuration of CPET-DL frameworks can be used to compromise the security of the CPET-hardened model.

## 9 Conclusion

We present ConPETro, the first systematic attack that exposes stealthy security risks through malicious configuration manipulation in CPET-DL frameworks. Our comprehensive evaluations across three widely adopted CPET-DL frameworks demonstrate ConPETro's effectiveness and highlight the limitations of current detection and defense mechanisms, which predominantly focus on plaintext models. Our findings reveal critical security gaps in CPET-hardened models, emphasizing the urgent need to develop robust CPET-DL protection mechanisms.

## Acknowledgement

The HKUST authors were supported in part by a RGC GRF grant under the contract 16214723, RGC CRF grant under the contract C6015-23G, and research fund provided by HSBC. We are grateful to the anonymous reviewers for their valuable comments.

## References

[1] [n. d.]. Research Artifact. https://sites.google.com/view/conpetro.
[2] 2023. EZKL. https://ezkl.xyz/.
[3] 2023. TF Encrypted. https://github.com/tf-encrypted/tf-encrypted/.
[4] 2024. Orion. https://github.com/gizatechxyz/orion.
[5] 2024. SecretFlow. https://github.com/secretflow/secretflow.
[6] Ehud Aharoni, Allon Adir, Moran Baruch, Nir Drucker, Gilad Ezov, Ariel Farkash, Lev Greenberg, Ramy Masalha, Guy Moshkowich, Dov Murik, Hayim Shaul, and Omri Soceanu. 2023. HeLayers: A Tile Tensors Framework for Large Neural Networks on Encrypted Data. *PETS*.
[7] Wei Ao and Vishnu Naresh Boddeti. 2024. AutoFHE: Automated Adaption of CNNs for Efficient Evaluation over FHE. In *USENIX Security'24*. 2173–2190.
[8] AWS. 2024. Cryptographic Computing - Amazon Web Services (AWS). https://aws.amazon.com/security/cryptographic-computing/.
[9] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *AISTATS*. PMLR.
[10] The Digital Banker. 2024. HSBC Fusion – AI Credit Assessment: Best credit assessment initiative - The Digital Banker. https://thedigitalbanker.com/hsbc-fusion-ai-credit-assessment-best-credit-assessment-initiative/.
[11] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2018. Scalable, transparent, and post-quantum secure computational integrity. (2018).
[12] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. arXiv:2104.03152 [cs.CR]
[13] Loris Bergerat, Anas Boudi, Quentin Bourgerie, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. 2023. Parameter optimization and larger precision for (T) FHE. *Journal of Cryptology* 36, 3 (2023), 28.
[14] Benjamin Bichsel, Samuel Steffen, Ilija Bogunovic, and Martin Vechev. 2021. DP-sniper: black-box discovery of differential privacy violations using classifiers. In *IEEE S&P*. IEEE, 391–409.
[15] Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. 2019. nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In *WAHC*. 45–56.
[16] Dan Bogdanov, Riivo Talviste, and Jan Willemson. 2012. Deploying Secure Multi-Party Computation for Financial Data Analysis: (Short Paper). In *FC*. Springer.
[17] Sean Bowe, Jack Grigg, and Daira Hopwood. 2019. Recursive proof composition without a trusted setup. *Cryptology ePrint Archive* (2019).
[18] Eric Brochu, Matthew W Hoffman, and Nando de Freitas. 2010. Portfolio allocation for Bayesian optimization. *arXiv preprint arXiv:1009.5419* (2010).
[19] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning web-scale training datasets is practical. In *IEEE S&P*. IEEE, 407–425.
[20] David Chaum, Claude Crépeau, and Ivan Damgard. 1988. Multiparty unconditionally secure protocols. In *STOC*. 11–19.
[21] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. 2024. Zkml: An optimizing system for ml inference in zero-knowledge proofs. In *EuroSys*. 560–574.
[22] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. 2019. A full RNS variant of approximate homomorphic encryption. In *SAC*. Springer, 347–368.
[23] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*. Springer, 409–437.
[24] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* 33, 1 (2020), 34–91.
[25] Neophytos Christou, Di Jin, Vaggelis Atlidakis, Baishakhi Ray, and Vasileios P Kemerlis. 2023. {IvySyn}: Automated vulnerability discovery in deep learning frameworks. In *USENIX Security'23*. 2383–2400.
[26] Eleanor Clifford, Ilia Shumailov, Yiren Zhao, Ross Anderson, and Robert Mullins. 2024. ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks. In *SaTML*. IEEE, 344–357.
[27] HEIR Contributors. 2023. HEIR: Homomorphic Encryption Intermediate Representation. https://github.com/google/heir.
[28] Pranav Dahiya, Ilia Shumailov, and Ross Anderson. 2024. Machine Learning needs Better Randomness Standards: Randomised Smoothing and {PRNG-based} attacks. In *USENIX Security'24*. 3657–3674.
[29] Valentin Dalibard, Michael Schaarschmidt, and Eiko Yoneki. 2017. BOAT: Building auto-tuners with structured Bayesian optimization. In *WWW*. 479–488.
[30] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. 2021. Lira: Learnable, imperceptible and robust backdoor attacks. In *ICCV*. 11966–11976.
[31] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).
[32] Diana Farrell, Fiona Greig, and Erica Deadman. 2020. Estimating Family Income from Administrative Banking Data: A Machine Learning Approach. *AEA Papers and Proceedings* 110 (May 2020), 36–41.
[33] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*. PMLR, 1180–1189.
[34] Yue Gao, Ilia Shumailov, and Kassem Fawaz. 2025. Supply-chain attacks in machine learning frameworks. In *MLSys*.

[35] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *STOC*. 169–178.

[36] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1985. The knowledge complexity of interactive proof-systems. In *STOC*. 291–304.

[37] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*. Springer, 305–326.

[38] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[39] Miguel Guevara. 2023. Expanding our Fully Homomorphic Encryption offering - Google Developers Blog. https://developers.googleblog.com/en/expanding-our-fully-homomorphic-encryption-offering/.

[40] Junfeng Guo, Ang Li, and Cong Liu. 2022. AEVA: Black-box Backdoor Detection Using Adversarial Extreme Value Analysis. In *ICLR*.

[41] Nima Shiri Harzevili, Jiho Shin, Junjie Wang, Song Wang, and Nachiappan Nagappan. 2023. Characterizing and understanding software security vulnerabilities in machine learning libraries. In *MSR*. IEEE, 27–38.

[42] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. 2019. Sok: General purpose compilers for secure multi-party computation. In *IEEE S&P*. IEEE, 1220–1237.

[43] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure Two-Party deep neural network inference. In *USENIX Security'22*. 809–826.

[44] Intel and Alibaba Cloud. 2022. Alibaba Builds End-to-End PPML Solution. https://www.intel.com/content/www/us/en/customer-spotlight/stories/alibaba-cloud-ppml-customer-story.html.

[45] ISACA. 2024. Exploring Practical Considerations and Applications for Privacy Enhancing Technologies. https://www.isaca.org/resources/white-papers/2024/exploring-practical-considerations-and-applications-for-privacy-enhancing-technologies.

[46] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE S&P*. IEEE, 19–35.

[47] Nazmul Karim, Abdullah Al Arafat, Adnan Siraj Rakin, Zhishan Guo, and Nazanin Rahnavard. 2024. Fisher information guided purification against backdoor attacks. In *CCS'24*. 4435–4449.

[48] B. Knott, S. Venkataraman, A.Y. Hannun, S. Sengupta, M. Ibrahim, and L.J.P. van der Maaten. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *arXiv 2109.00984*.

[49] Torsten Krauß and Alexandra Dmitrienko. 2023. Mesas: Poisoning defense for federated learning resilient against adaptive attackers. In *CCS'23*. 1526–1540.

[50] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[51] Harry Langford, Ilia Shumailov, Yiren Zhao, Robert Mullins, and Nicolas Papernot. 2025. Architectural neural backdoors from first principles. In *IEEE S&P*. IEEE.

[52] Ryan Lavin, Xuekai Liu, Hardhik Mohanty, Logan Norman, Giovanni Zaarour, and Bhaskar Krishnamachari. 2024. A Survey on the Applications of Zero-Knowledge Proofs. *arXiv preprint arXiv:2408.00243* (2024).

[53] Seewoo Lee, Garam Lee, Jung Woo Kim, Junbum Shin, and Mun-Kyu Lee. 2023. HETAL: efficient privacy-preserving transfer learning with homomorphic encryption. In *ICML*. JMLR.org, Article 786, 26 pages.

[54] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. 2021. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *ICSE*. IEEE, 263–274.

[55] Yichen LI, Dongwei Xiao, Zhibo Liu, Qi Pang, and Shuai Wang. 2024. Metamorphic Testing of Secure Multi-Party Computation (MPC) Compilers. In *ESEC/FSE*.

[56] Chao Lin, Min Luo, Xinyi Huang, Kim-Kwang Raymond Choo, and Debiao He. 2021. An efficient privacy-preserving credit score system based on noninteractive zero-knowledge proof. *IEEE systems journal* 16, 1 (2021), 1592–1601.

[57] Yehuda Lindell. 2020. Secure multiparty computation. *Commun. ACM* 64, 1 (2020), 86–96.

[58] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*. Springer, 182–199.

[59] Forbes Media LLC. 2024. AI's Role In Modern Medical Diagnosis. https://www.forbes.com/councils/forbesbusinesscouncil/2024/10/14/ais-role-in-modern-medical-diagnosis/.

[60] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Cheng Hong, Kui Ren, Tao Wei, and WenGuang Chen. 2023. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive* (2023).

[61] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Wenjing Fang, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. 2023. SecretFlow-SPU: A performant and User-Friendly framework for Privacy-Preserving machine learning. In *ATC'23*. 17–33.

[62] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[63] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–35.

[64] Sarah Meiklejohn, Hayden Blauzvern, Mihai Maruseac, Spencer Schrock, Laurent Simon, and Ilia Shumailov. 2025. Machine Learning Models Have a Supply Chain Problem. *arXiv preprint arXiv:2505.22778* (2025).

[65] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference system for neural networks. In *PPMLP*. 27–30.

[66] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *CVPR*. 1765–1773.

[67] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *DSS* 62 (2014), 22–31.

[68] Christian Mouchet, Sylvain Chatel, Apostolos Pyrgelis, and Carmela Troncoso. 2024. Helium: Scalable MPC among lightweight participants and under churn. In *CCS*. 3038–3052.

[69] Kundan Munjal and Rekha Bhatia. 2023. A systematic review of homomorphic encryption and its contributions in healthcare industry. *Complex & Intelligent Systems* 9, 4 (2023), 3759–3786.

[70] Tuomas Oikarinen and Diego Dorn. 2023. Training a NN to 99% accuracy on MNIST in 0.76 seconds. https://github.com/tuomaso/train_mnist_fast.

[71] Qi Pang, Yuanyuan Yuan, and Shuai Wang. 2024. MPCDiff: Testing and Repairing MPC-Hardened Deep Learning Models. In *NDSS*.

[72] Jaiyoung Park, Michael Jaemin Kim, Wonkyung Jung, and Jung Ho Ahn. 2022. AESPA: Accuracy preserving low-degree polynomial activation for fast private inference. *arXiv preprint arXiv:2201.06699* (2022).

[73] A Paszke. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[74] Hongwu Peng, Shaoyi Huang, Tong Zhou, et al. 2023. Autorep: Automatic relu replacement for fast private network inference. In *ICCV*. 5178–5188.

[75] Yiteng Peng, Daoyuan Wu, Zhibo Liu, Dongwei Xiao, Zhenlan Ji, Juergen Rahmel, and Shuai Wang. 2025. Testing and Understanding Deviation Behaviors in FHE-Hardened Machine Learning Models. In *ICSE*. IEEE, 2251–2263.

[76] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*. PmLR, 8748–8763.

[77] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2020. POSEIDON: Privacy-preserving federated neural network learning. *arXiv preprint arXiv:2009.00349* (2020).

[78] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *IEEE S&P*. IEEE.

[79] Andrei Stoian, Jordan Frery, Roman Bredehoft, Luis Montero, Celia Kherfallah, and Benoit Chevallier-Mames. 2023. Deep neural networks for encrypted inference with TFHE. In *CSCML*. Springer, 493–500.

[80] Bing Sun, Jun Sun, Wayne Koh, and Jie Shi. 2024. Neural Network Semantic Backdoor Detection and Mitigation: A Causality-Based Approach. In *USENIX Security'24*.

[81] Haochen Sun, Jason Li, and Hongyang Zhang. 2024. zkllm: Zero knowledge proofs for large language models. In *CCS*. 4405–4419.

[82] Guanhong Tao, Shengwei An, Siyuan Cheng, Guangyu Shen, and Xiangyu Zhang. 2023. Hard-label black-box universal adversarial patch attack. In *USENIX Security'23*. 697–714.

[83] Alexander Viand, Patrick Jattke, and Anwar Hithnawi. 2021. SoK: Fully homomorphic encryption compilers. In *IEEE S&P*. IEEE, 1092–1108.

[84] A Helen Victoria and Ganesh Maragatham. 2021. Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems* 12, 1 (2021), 217–223.

[85] Hang Wang, Zhen Xiang, David J Miller, and George Kesidis. 2024. Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic. In *IEEE S&P*. IEEE, 1994–2012.

[86] Yuxin Wang, Zeyu Ding, Daniel Kifer, and Danfeng Zhang. 2020. Checkdp: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In *CCS*. 919–938.

[87] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. 2022. PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier. In *USENIX Security'22*. 2065–2082.

[88] Dongwei Xiao, Zhibo Liu, Yiteng Peng, and Shuai Wang. 2025. MTZK: Testing and Exploring Bugs in Zero-Knowledge (ZK) Compilers. In *NDSS*.

[89] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[90] Zama. 2022. Concrete ML: a Privacy-Preserving Machine Learning Library using Fully Homomorphic Encryption for Data Scientists. https://github.com/zama-ai/concrete-ml.

[91] Zama. 2024. Concrete ML model builds quantization parameters based on the data ranges. https://community.zama.ai/t/simple-linreg-model-outputs-wrong-results/902.