SoK: A Taxonomy of Attacks and Defenses in Split Learning

Aqsa Shabbir[1][0009−0000−0331−9648]⋆, Halil İbrahim Kanpak[2][0009−0005−0284−5941]⋆, Alptekin Küpçü[2,3][0000−0003−2099−2206]⋆⋆, and Sinem Sav[1][0000−0001−9096−8768]⋆⋆

[1] Department of Computer Engineering, Bilkent University, Ankara, Türkiye
[2] Department of Computer Engineering, Koç University, İstanbul, Türkiye
[3] KUIS AI Center, Koç University, İstanbul, Türkiye

Abstract. Split Learning (SL) has emerged as a promising paradigm for distributed deep learning, allowing resource-constrained clients to offload portions of their model computation to servers while maintaining collaborative learning. However, recent research has demonstrated that SL remains vulnerable to a range of privacy and security threats, including information leakage, model inversion, and adversarial attacks. While various defense mechanisms have been proposed, a systematic understanding of the attack landscape and corresponding countermeasures is still lacking. In this study, we present a comprehensive taxonomy of attacks and defenses in SL, categorizing them along three key dimensions: employed strategies, constraints, and effectiveness. Furthermore, we identify key open challenges and research gaps in SL based on our systematization, highlighting potential future directions.

Keywords: split learning · collaborative learning · distributed machine learning · privacy-preserving computation · data privacy.

1 Introduction

The increasing adoption of Machine Learning (ML) as a Service has revolutionized artificial intelligence-driven applications across various domains, including healthcare, finance, and the Internet of Things (IoT). However, ML outsourcing raises privacy concerns, as sensitive user data is often processed on external or untrusted servers [65]. To address this challenge, Privacy-Preserving Machine Learning (PPML) techniques have been developed, enabling collaborative learning while ensuring data confidentiality and privacy.

Among various collaborative methods, Split Learning (SL), which partitions deep learning models between clients and servers [23], emerged as a promising framework. SL enables clients to process several layers of a neural network locally while delegating the remaining layers to a server. This design offloads computation from resource-constrained clients while keeping raw data local [73]. SL

⋆ These authors contributed equally to this work.
⋆⋆ Corresponding authors. akupcu@ku.edu.tr, sinem.sav@cs.bilkent.edu.tr

has evolved into various architectures, e.g., Vanilla Split Learning (VanSL) [23] or U-shaped Split Learning. Despite its architectural advantages for resource-constrained clients and inherent data locality [23, 73], SL faces multiple security and privacy risks. Data reconstruction attacks exploit smashed data and gradients to infer private inputs, while label inference attacks analyze gradient updates to recover class labels [17, 54]. Property inference attacks extract sensitive attributes without full data reconstruction [49], and model inversion [10] aims to recover input samples from model activations. Additionally, backdoor and poisoning attacks [4, 81] manipulate the training pipeline to implant adversarial behavior. These vulnerabilities expose critical gaps in SL, necessitating robust defense mechanisms to mitigate potential threats. However, before effective defenses can be designed and evaluated, a systematic understanding of the *attack landscape* itself is required. We build this foundation by addressing core research questions regarding *attacks* against SL:

RQ1: *How can adversarial objectives targeting SL be systematically taxonomized? Which architectural vulnerabilities serve as primary attack vectors?*

RQ2: *Which core characteristics define SL attack mechanisms (e.g., feature-space hijacking, model inversion, gradient-based inference, embedding poisoning) in terms of their ability to generalize across different SL variants and data domains, their potential for stealth, and their capacity to circumvent defenses?*

RQ3: *Under what operational constraints and adversarial assumptions (regarding knowledge, access, capabilities, and collusion scenarios) do attacks against SL demonstrate measurable effectiveness? How can their quantifiable impact on privacy and model integrity be systematically evaluated?*

To counter these threats, researchers have explored various privacy-preserving techniques as defense strategies, including Differential Privacy (DP) to perturb gradients [79], Homomorphic Encryption (HE) for encrypted computation to prevent unauthorized data access [56], and communication compression to reduce information leakage [57]. While each method strengthens SL security, it also introduces trade-offs in computational overhead, scalability, and model performance. Therefore, a clear understanding of the defense landscape, its capabilities, limitations, and evaluation is equally critical. Consequently, this paper also addresses these research questions concerning *defenses* in SL:

RQ4: *What is the taxonomy of defense techniques against predefined attack scenarios for SL? What are the common techniques and tools being employed?*

RQ5: *Which defenses are suitable for a specific task? What are the constraints and conditions implied by these defenses? Can these techniques be optimized? Are these techniques suitable for real-world applications?*

RQ6: *How do defense techniques achieve effectiveness across different scenarios with respect to preserving the confidentiality and integrity of training? What criteria define a successful defense mechanism?*

This paper provides a comprehensive SL analysis by systematically categorizing attack vectors and defense strategies based on these research questions. We propose taxonomies for attacks and defenses, evaluating their methodology, constraints, and effectiveness. We highlight key trade-offs, our observations, and

open challenges derived from this systematization, paving the way for future advancements in SL privacy and robustness.

2 Review Scope and Methodology

We systematically categorize SL architectures, identify key attack surfaces—such as data reconstruction, feature leakage, and adversarial interference—and evaluate defenses including DP, HE, and adversarial training. By synthesizing prior work, we offer a structured framework to assess privacy risks, defense efficacy, and security trade-offs, guiding future progress in PPML.

Search Methodology. To ensure a comprehensive and representative collection of research on SL, we systematically gathered papers from major academic databases, including Google Scholar, IEEE Xplore, ACM Digital Library, arXiv, and SpringerLink. The search was conducted in two stages: an automated search using predefined keyword queries, followed by a snowballing approach to refine and expand the selection of relevant studies. We used keyword combinations to capture diverse SL research, including privacy, attack surfaces, and defenses. Example keywords included a combination of "*split learning*", "*privacy-preserving*", "*split neural networks*", "*privacy attacks on split learning*", "*homomorphic encryption based split learning*", "*differential privacy based split learning*", and "*inference attacks in split learning*". These queries targeted research addressing both theoretical and practical aspects of SL, ensuring coverage of multiple perspectives on its security vulnerabilities and mitigation strategies. Then, using a snowballing approach, we examined references of key papers and their citations, enabling us to trace the field's development over time.

After identifying relevant works, we aligned and classified them to build our taxonomy for SL systematization. Each paper was first screened by its abstract, then examined in detail if relevant techniques were indicated. We excluded papers not explicitly focused on SL, such as those on general distributed or outsourced learning. Each selected work was classified across multiple dimensions, including proposed defenses (for attack papers), addressed attacks (for defense papers), threat models, and server-client configurations.

Systematization Methodology. For our systematization, we begin by reviewing SL architectures to establish a foundational understanding of the landscape. We then categorized the collected papers into two primary groups: attack papers and defense papers. Within each group, we employ a structured classification approach based on three key dimensions:

1. Strategies: The strategies used to conduct attacks or implement defenses.
2. Constraints: The assumptions, limitations, and requirements under which the attacks or defenses operate.
3. Effectiveness: The impact and success rate of attacks, as well as the robustness and trade-offs of defense mechanisms.

Finally, Fig. 1 presents a timeline summarizing the surveyed attacks and defenses, highlighting their chronological progression and dominant strategies.

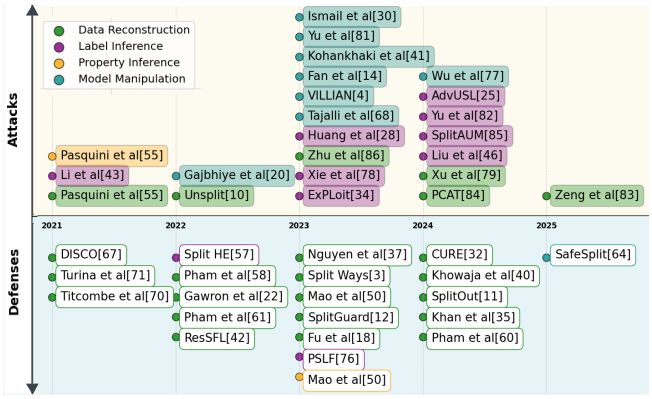

Fig. 1. SL attack and defense timeline (2021–2025), with attacks on the top half and defenses on the bottom half. Attack and defense types are categorized into four groups, as shown in the legend. Attacks are represented with a colored background, while the corresponding defenses share the same color but are displayed without a background.

3 Split Learning (SL)

Here, we present an overview of split learning (SL), its key concepts, and variants. SL [23] is a distributed learning paradigm that splits model computation between clients and servers. It has evolved into several variants, which we outline below and illustrate the most common ones in Fig.2.

Vanilla Split Learning (VanSL): VanSL is the simplest SL architecture, where the client computes the initial layers (f_c) and the server handles the remaining layers (f_s). The client uses its private data x to compute the *intermediate activations* $z_c = f_c(x)$, which are transmitted to the server. The server completes the *forward pass* by calculating the output $\hat{y} = f_s(z_c)$. During backpropagation (f'), the server computes the gradient of the loss function L with respect to z_c, denoted as $\nabla_{z_c} = \frac{\partial L}{\partial z_c}$, and sends it to the client. The client then updates the parameters of f_c, ensuring that x remains on the client side.

U-shaped Split Learning (USL): USL divides the model into three segments: (i) *initial layers* f_c processed by the client, (ii) *middle layers* f_s handled by the server, and (iii) *remaining layers* f_{c_r} completed by the client. The client begins by processing x through f_c to compute z_c and sends z_c to the server. The server processes z_c through f_s to compute z_s and returns z_s to the client. The client completes the forward pass by applying f_{c_r} to z_s, resulting in the final output $\hat{y}$. Backpropagation mirrors this process in reverse. USL thus enables end-to-end training while keeping both x and $\hat{y}$ on the client side.

Hybrid Split Learning: Hybrid-SL integrates SL with other collaborative learning paradigms, such as Federated Learning (FL) to enhance privacy, scalability, and efficiency. For instance, combining SL with FL, Split Federated Learning (SFL) allows clients to train parts of the model collaboratively while leveraging federated learning aggregation techniques. As illustrated in the SFL part

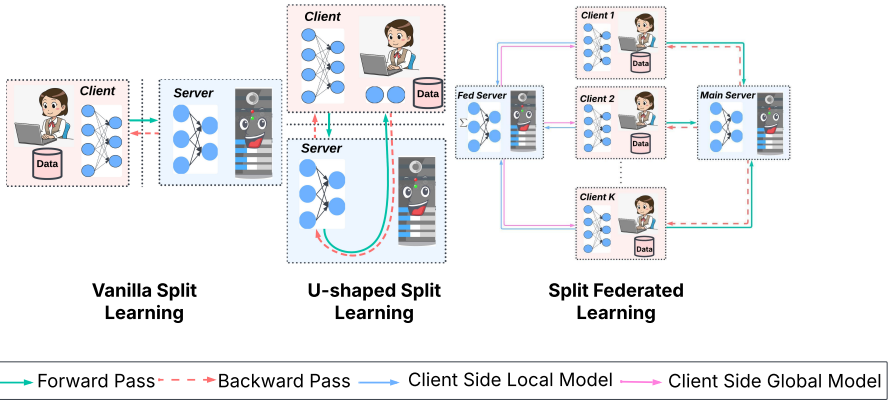

Fig. 2. Overview of three key SL variants: Vanilla Split Learning, U-shaped Split Learning, and Split Federated Learning. The figure highlights the architectural distinctions, client-server roles, and the flow of forward and backward passes in training.

of Fig. 2 (third sub-figure), clients compute the initial layers locally and send the smashed data to the Fed Server, which processes intermediate layers and forwards the activations to the Main Server. During backpropagation, gradients return from the Main Server to the Fed Server and then to the clients, after which the Fed Server aggregates the client-side updates (e.g., via FedAvg) before synchronizing the global model. Similarly, incorporating HE ensures that smashed data exchanged between clients and servers remains encrypted, further strengthening privacy. It enables solutions for scenarios involving heterogeneous clients and large-scale data distributions. However, integrating additional mechanisms often introduces computational and communication overhead, making efficient design and implementation critical for practical use.

Multi-hop Split Learning (MHSL): MHSL extends SL across multiple parties, forming a computation chain. The model is divided into several segments $(f_{c_1}, f_{s_1}, f_{s_2}, ..., f_{c_n})$ where the subscript denotes the entity processing that segment. The client with input x begins by computing $(z_{c_1} = f_{c_1}(x))$ and sends z_{c_1} to the first server. Each subsequent server i computes $z_{s_i} = f_{s_i}(z_{s_{i-1}})$ and forwards it to the next entity. The final client computes the output $\hat{y} = f_{c_n}(z_{s_{n-1}})$ and calculates the loss. During backpropagation, gradients flow in the reverse direction through the chain. MHSL allows flexible client-server ordering, with practical limits: the first client processes x, and the final party computes the loss. Intermediate roles vary by need—e.g., hospitals may handle endpoints in medical use, while cloud servers process the middle; in business forecasting, companies may preprocess data before server-side inference. This flexibility supports a range of privacy-preserving federated learning scenarios, where multiple entities jointly contribute to the computation while maintaining their data autonomy.

No-Label Split Learning (NLSL): NLSL keeps both x and y private to the client. The client processes f_c to compute z_c and sends it to the server. The

server computes the forward pass through the remaining layers f_s to generate $\hat{y}$ and sends it to the client. The client then computes the loss $L(y, \hat{y})$ using the true labels y and calculates the gradient ($\nabla_{\hat{y}} = \frac{\partial L}{\partial \hat{y}}$). The client sends $\nabla_{\hat{y}}$ back to the server, which begins the backpropagation process by computing ($\nabla_{z_c} = \frac{\partial L}{\partial z_c}$) and sends it to the client. Finally, the client updates the parameters of f_c using the received gradients. Similar to USL, NLSL ensures the client computes the loss, preventing the server from accessing y. Unlike USL, however, the server completes the full forward pass, with only the client handling label-related tasks.

3.1 Split Learning Partitioning Schemes

SL's effectiveness also depends on the data partitioning scheme which determines how data is distributed among clients, impacting both the training process and privacy assurances. In **Horizontal Split Learning (HSL)** [43,73], clients handle data with identical feature spaces but different samples; for instance, multiple hospitals with similar patient data but distinct patient groups can collaboratively train a model by sharing intermediate activations without exposing raw data. Conversely, **Vertical Split Learning (VSL)** [30,73] is applicable when clients possess complementary features for the samples; each client processes its unique features, and a server combines these embeddings to facilitate joint learning, such as a bank and an e-commerce platform sharing distinct user attributes to jointly develop a credit scoring model.

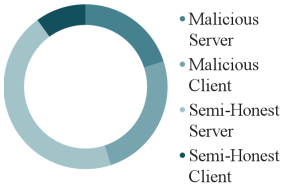

Fig. 3. The distribution of adversarial models of surveyed attacks in SL.

3.2 Client-Server Participation Models

Here, we define SL's client-server participation models, detailing the collaboration dynamics between multiple clients and servers. It determines how the model is split and how updates are shared.

Single Client Split Learning (SCSL) In SCSL [72], training involves one client and a server, making it ideal for a single organization, such as a hospital or research institute, that seeks to preserve privacy while offloading the heavy computation to a server. Although it simplifies communication and coordination, clients still need sufficient resources to process their local model portions.

Multi Client Split Learning (MCSL) MCSL [60] keeps the forward and backward propagation split between multiple clients and a server. Clients only

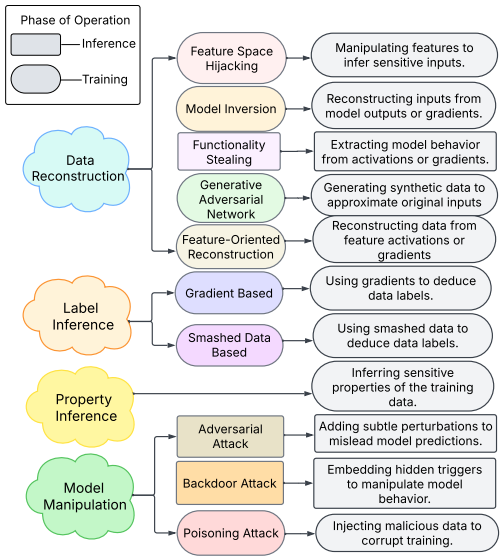

Fig. 4. Taxonomy of SL attacks with four main vectors: (1) Data Reconstruction, (2) Label Inference, (3) Property Inference, and (4) Model Manipulation Attacks. The shape convention indicates the phase of attack: Ovals (◯) represent training-phase attacks, rectangles (□) represent inference-phase attacks.

send smashed data to the server, and the server processes the deeper layers of the model and updates all clients.

Single Server Split Learning (SSSL) In SSSL [73], all clients send intermediate activations to a single server, which completes forward and backward passes before returning gradients. SSSL relies on a centralized server, simplifying coordination and system design. This is ideal for organizations with strong infrastructure, but scalability may decline as the number of clients increases.

Multi-Server Split Learning (MSSL) MSSL [60] distributes model computation across multiple servers, improving scalability and resource utilization. Clients process initial layers and send smashed data to servers, which partition and collaboratively handle the remaining computations. MSSL suits large-scale deployments where a single server is insufficient, offering better fault tolerance.

4 Attacks on Split Learning

We categorize SL threats across three dimensions: *attack strategies* (Section 4.1), *operational constraints* (Section 4.2), and *attack effectiveness* (Section 4.3).

4.1 Attack Strategies

We classify SL attacks into four main strategies: data reconstruction attacks, label inference attacks, property inference attacks, and model manipulation at-

tacks. Each strategy manifests either in the *training phase*, where adversaries exploit gradients, smashed activations, or updates, or in the *inference phase*, where adversaries exploit model predictions or query access. We summarize the taxonomy of attacks together with their phase of operation in Fig. 4, which also guides the ordering of this section. Furthermore, the distribution of adversarial models across these strategies is summarized in Fig. 3.

Data Reconstruction Attacks Data Reconstruction Attacks represent a significant privacy vulnerability in SL systems, wherein the adversary attempts to recover sensitive training data. These attacks leverage model characteristics, including gradients, confidence scores, and internal representations, to reconstruct private training inputs. The adversary optimizes a reconstruction function to approximate the original input x through $\tilde{x} = \arg\min_{\tilde{x}^*} \mathcal{L}_{\text{rec}}(f_c(\tilde{x}^*), z_c) + \lambda R(\tilde{x}^*)$ where $\tilde{x}$ is the reconstructed input, and $\tilde{x}^*$ is iteratively refined to align with x. The reconstruction loss $\mathcal{L}_{\text{rec}}$ minimizes the difference between the reconstructed and original feature representations. At the same time, regularization $R(\tilde{x}^*)$ imposes constraints such as Total Variation (TV) loss [64] or adversarial regularization [50]. The term λ controls the balance between accuracy and constraint enforcement. Their effectiveness is typically evaluated using standard reconstruction fidelity metrics. Mean Squared Error (MSE) measures pixel-level deviations between reconstructed and original images, Peak Signal-to-Noise Ratio (PSNR) quantifies reconstruction quality relative to distortion, and Structural Similarity Index (SSIM) assesses perceptual similarity in terms of luminance, contrast, and structure. Together, these metrics provide a compact yet comprehensive indication of reconstruction quality and the extent of information leakage. The primary reconstruction methodologies are detailed below:

Feature Space Hijacking Attack (FSHA). FSHA [21, 54] is a data reconstruction attack that exploits feature representations in SL. By leveraging adversarial learning, FSHA enables the adversary to approximate private client data from the exchanged feature space. This attack employs a three-component system: a pilot network ($\tilde{f}_c$) that defines the target feature space, an inverse network ($\tilde{f}_c^{-1}$) that reconstructs inputs from features, and a discriminator (D) that guides mapping learning through adversarial training. Here, $\tilde{f}_c$ generates synthetic feature embeddings $\tilde{z_c}$ that $\tilde{f}_c^{-1}$ learns to invert, while D aligns $\tilde{z_c}$ with the z_c. Evaluation results [54] show that FSHA quickly reduces reconstruction error: on the MNIST dataset, MSE starts around 0.18–0.20 and drops below 0.05 within the first 2000 iterations, eventually converging toward approximately 0.01 by 10,000 iterations, demonstrating fast and high-fidelity recovery of x.

Model Inversion. An adversary attempts to reconstruct private training data by exploiting the model's parameters, outputs, or gradients. As demonstrated by [10, 25], a malicious party can leverage the information received during SL to recover client data. This approach uses an optimization process in which the adversary generates synthetic inputs that reproduce the observed training activations or gradients. By minimizing the difference, typically via MSE, between these synthetic outputs and the real ones, the adversary can approximate the

original private data. The authors achieve MSE values of 0.048 on MNIST and 0.084 on F-MNIST when attacking trained models, illustrating that even limited-information adversaries can obtain semantically faithful reconstructions [10].

Functionality Stealing. In functionality stealing [83], an adversary replicates a model's behavior to approximate its functionality and extract sensitive information. The attacker can train a pseudo-client model ($\tilde{f}_c$) that mimics the outputs of the original model f_c, without access to its internal architecture. After stealing functionality, the adversary can train a reverse mapping to reconstruct x, compromising privacy across clients. Empirical results show that the $\tilde{f}_c$ preserves the victim model's behavior with only a small accuracy gap (e.g., 98.3% vs. 99.1% on MNIST) and enables reconstructions with low error (MSE < 0.02), confirming that the stolen functionality suffices to leak identifiable input information.

Generative Adversarial Network (GAN). GANs have emerged as powerful tools for data reconstruction attacks [49,82]. This approach involves training the generator G to produce synthetic inputs that yield feature representations or gradients matching those observed during training. The discriminator D helps refine these reconstructions by providing feedback on their realism. GAN-based attacks are especially concerning due to their ability to improve over time and operate with limited information. For example, [82] achieves effective reconstructions on CIFAR100 using only 1% auxiliary data, and reconstructed samples exhibit high semantic fidelity even when pixel-level accuracy is imperfect.

Feature Reconstruction. Feature reconstruction aims to recover the original input data from intermediate feature representations exchanged during model training. Using statistical metrics [22, 46] and adversarial learning, attackers can infer private data by exploiting representation biases in z_c [78, 85]. Evaluations [78] show high fidelity at shallow splits (e.g., SSIM ≈ 0.93, PSNR ≈ 25.9 at layer 1) and still identifiable structure at deeper ones (SSIM ≈ 0.62), confirming that smashed features retain substantial recoverable content.

Label Inference Attacks Label inference attacks exploit the correlation between model updates and labels to infer client information. By analyzing shared gradients or smashed data during training, these attacks leverage inherent patterns in the learning process to reconstruct labels. They are primarily categorized into gradient-based and smashed data-based label inferences. Their effectiveness is typically measured using label-recovery accuracy, precision, recall, and confusion-matrix analysis.

Gradient-Based Label Inference. These attacks exploit the correlation between gradient updates and label distributions to infer client information. Three methods dominate the literature for performing gradient-based label inference attacks: (i) similarity-based techniques [42,45] that compare gradients directly by selecting the expected label (y_{exp}) and minimizing the difference between the observed gradient (∇_c) and expected gradient (∇_{exp}) through $\tilde{y} = \arg\min_{y_{\text{exp}}} \|\nabla_c - \nabla_{exp}\|^2$ where $\|\cdot\|$ represents the similarity metric, such as Euclidean norm (ℓ_2-norm). (ii) loss function-based inference [10, 77] that refines predictions in searching for the most expected label y_{exp} by minimizing MSE or cross-entropy

loss: $\tilde{y} = \arg\min_{y_{\text{exp}}} \mathcal{L}_{\text{adv}}(\nabla_c - \nabla_{exp})$ where $(\mathcal{L}_{\text{adv}})$ is the adversary loss function measuring the difference between ∇_c and ∇_{exp}, and (iii) surrogate model optimization [4,33,84], where an adversary iteratively refines the estimated label by minimizing the loss function using gradient-based optimization. Unlike direct gradient matching or similarity-based inference, this method leverages a surrogate model to approximate the relationship between x, ∇, and y, allowing for a structured reconstruction of private labels. At each iteration t, the adversary updates the estimated label $\tilde{y}(t)$ by minimizing the loss function:

$$\tilde{y}^{(t+1)} = \tilde{y}^{(t)} - \eta \frac{\partial \mathcal{L}\text{adv}(\nabla_c, \nabla exp(\tilde{y}^{(t)}))}{\partial \tilde{y}} \tag{1}$$

where η controls the step size for updating the label estimate. Empirically, [33] achieve high leakage reporting label-recovery accuracies of 99.53% on CIFAR-10, 94.38% on CIFAR-100, and 80.61% on Tiny-ImageNet.

Smashed Data-Based Label Inference. An adversary aims to reconstruct private labels by analyzing the structural and semantic properties of z received by the adversary. Unlike gradient-based attacks, these methods do not require gradient updates but instead rely on the inherent information encoded in z. The adversary infers labels by minimizing a predefined loss function $(\mathcal{L}_{\text{adv}})$, which measures the difference between the observed z and reference embeddings (z_{exp}). This process is formulated as: $\tilde{y} = \arg\min_{y_{\text{exp}}} \mathcal{L}_{\text{adv}}(\mathcal{F}(z), \mathcal{F}(z_{\text{exp}}))$ where, $\mathcal{F}$ represents the adversarial function. Researchers have identified three main approaches for label inference from smashed data in SL: (i) distance-based matching [45] where the adversary assigns z_c to the nearest z_{exp} using a metric such as Euclidean distance, (ii) clustering [45, 85] where z_c are grouped into class-coherent clusters (e.g., via K-Means) and labels are inferred from the majority label of the cluster, and (iii) transfer learning [27,45] applies a pre-trained model to transform z_c into a more separable feature space before matching. The quantitative results in [45] indicate that smashed-data attacks remain highly effective even without gradients, with accuracy ranging from approximately 0.30 on ImageNet to above 0.90 on datasets such as Fruits-360 and F-MNIST.

Observation. ***Inadequate Formalization of Smashed Data:*** *In SL, the smashed data retains semantic features strongly correlated with the input x. While studies have demonstrated reconstruction [49, 54] and label inference attacks [45, 85] from z, no standardized metric yet exists to evaluate semantic vulnerability. This gap directly pertains to RQ2, which seeks to characterize the nature and origins of information leakage in SL.*

Open Problem. ***Toward Theoretical Bounds for Smashed Data Leakage:*** *A key research gap is the absence of theoretical frameworks to quantify the semantic information retained in smashed data. Existing SL defenses are ad hoc, lacking metrics to assess their robustness against attacks. Formalizing the link between mutual information $I(x, z)$ and attack success could enable provably secure cut-layer designs or compression schemes with bounded leakage—crucial for SL systems in high-stakes domains like healthcare and finance.*

Property Inference Attacks Property inference attacks extract sensitive attributes or statistical patterns from client data. Unlike reconstruction attacks, the aim is to infer specific properties such as demographic information or class distributions. Property inference attacks in SL typically leverage the intermediate representations or model updates shared during training [49, 54]. The adversary trains a classifier C to map smashed data or predictions to property labels, inferring unintended private attributes. The attack is formulated as:

$$A_{PIA} = \arg\min_{F} \sum_{x_i \in X} \mathcal{L}_{adv}(F(T(x_i)), l_i), \quad l_i \in \{0, 1\} \tag{2}$$

where (F) represents the adversarial inference model, (T) is the observable outputs during inference, and (l_i) is the property being inferred. These attacks are particularly concerning because they can reveal sensitive information while the participating entities believe they are only sharing task-relevant features. Results [54] reveals that property inference can be highly effective achieving over 90% gender-prediction accuracy on CelebA and UTKFace.

Observation. ***Generalization of Attack Mechanisms:*** *Cut-layer attacks, such as latent representation reconstruction [49, 54] and property inference [54], succeed across diverse SL models and datasets. This indicates that the act of transmitting z introduces structural privacy risks that persist regardless of task or architecture. These findings support RQ2, underscoring the need for model-agnostic defenses against leakage from both z and ∇.*

Observation. ***Vulnerability of the Cut Layer:*** *The exchange point between client and server, where z or ∇ are transmitted, consistently serves as a key attack surface in SL. A range of attacks exploit this interface to extract sensitive information, indicating that obfuscating raw inputs alone is insufficient, such as local DP. This underscores the need for defenses that specifically target information leakage at the cut layer, as highlighted in RQ2.*

Model Manipulation Attacks Here, we explore attacks that compromise the integrity of SL model, categorizing them into: (i) adversarial attacks that perturb inputs to cause misclassification, and (ii) backdoor and poisoning attacks that alter training data to induce malicious behavior or degrade performance. Their effectiveness is typically measured by accuracy drop and misclassification rate to capture performance degradation; perturbation magnitude (ϵ) to bound adversarial distortion; and Attack Success Rate (ASR) to quantify the proportion of successful manipulations. For backdoor and poisoning attacks, ASR is evaluated alongside Clean Data Accuracy (CDA), per-class recall degradation, and the poisoning ratio to reflect the proportion of tampered training samples.
Adversarial Attacks. Adversarial attacks are predominantly inference phase attacks where the adversary manipulates intermediate feature representations by introducing perturbations at test time that induce misclassification. These attacks can be categorized as non-targeted attacks [14] and targeted attacks [24]. The objective of a non-targeted attack is to maximize the difference between the

perturbed representation and the clean feature representation z. This is achieved by often using cosine similarity or Euclidean distance:

$$\xi^* = \arg \max_{\|\xi\|_\infty \leq \epsilon} \mathcal{L}_{\text{attack}}(z, \xi + z) \tag{3}$$

where ξ^* is the optimal adversarial perturbation, ξ is the adversarial perturbation applied to z, and $\|\xi\|_\infty \leq \epsilon$ ensures that the perturbation remains within the allowed bound. However, in a targeted attack, the adversary modifies z such that they move toward a predefined target embedding z_t, forcing the model to produce a specific incorrect prediction. This is achieved by minimizing (rather than maximizing) the loss function in Equation 3. The reason is that in a non-targeted attack, the adversary aims to *maximize* the deviation of the perturbed representation from the original feature representation, making the output unpredictable and unreliable. In contrast, a targeted attack seeks to *minimize* the difference between the perturbed representation and a specific target representation z_t, effectively steering the model toward controlled misclassifications. Empirical results [14] show that perturbations substantially degrade performance, causing 16–30% accuracy drops on CIFAR-10 and up to 34% on CIFAR-100. Even with only 128 proxy samples, the attack still induces a 14–20% accuracy drop.

Backdoor and Poisoning Attacks. SL models are vulnerable to adversarial manipulations that exploit weaknesses in the training process, particularly through poisoning attacks and backdoor attacks. *Poisoning attacks* degrade model performance by modifying the training dataset. Let x, y denote the training data and labels, and $\mathcal{A}(x, y)$ be the adversarial poisoning function that generates a poisoned x', y'. The model, parameterized by θ, updates its parameters to θ^* after training on poisoned data, leading to incorrect decision boundaries. The adversary aims to increase classification errors, causing the model to minimize its loss in a way that harms generalization and boosts misclassification:

$$\theta^* = \arg \min_\theta \mathbb{E}_{(x',y') \sim \mathcal{A}(x,y)} \mathcal{L}(f_\theta(x'), y') \tag{4}$$

where $f_\theta(x')$ is the model output, and $\mathcal{L}(\cdot)$ measures the classification loss. The expected loss $\mathbb{E}_{(x',y') \sim \mathcal{A}(x,y)}$ ensures the generalization across training samples.

Backdoor attacks implant hidden triggers in the training process, enabling adversaries to manipulate model predictions when the trigger is activated in the inference phase while maintaining normal behavior otherwise. Let x_b be the backdoor-inserted inputs with the attacker-defined target labels y_t. The adversary optimizes a dual-objective function that ensures normal classification on clean samples while inducing misclassification on backdoor samples:

$$\theta^* = \arg \min_\theta \mathbb{E}_{(x,y) \sim \mathcal{D}_{\text{clean}}} \mathcal{L}(f_\theta(x), y) + \lambda \mathbb{E}_{(x_b,y_t) \sim \mathcal{D}_{\text{b}}} \mathcal{L}(f_\theta(x_b), y_t)$$

where $\mathcal{D}_{\text{clean}}$ is the clean dataset, while $\mathcal{D}_{\text{b}}$ contains samples modified with a trigger. The adversary applies a backdoor function to x, producing x_b that, when processed by the model, leads to a specific misclassification. The attacker assigns target labels y_t to these backdoor-embedded samples, ensuring that normal inputs retain their correct classification. λ regulates the trade-off between accuracy

on clean inputs and the integration of backdoor functionality. We further detail these attacks into subcategories, i.e., label flipping, embedding poisoning, client-side backdoor, and server-side backdoor attacks in Appendix A.

Observation. ***Semi-Honest Adversaries:*** *Our taxonomy reveals a critical reliance on the semi-honest adversarial model. While this simplifies threat modeling, it underestimates real-world risks, as attacks like label flipping, gradient manipulation, or collusion exceed the assumptions of this model, rendering many defenses ineffective against stronger adversaries per RQ3.*

Open Problem. ***Malicious and Colluding Adversaries:*** *Current SL defenses lack rigorous evaluation against malicious, covert, and colluding adversaries, leaving a gap in both threat modeling and empirical validation.*

Observation. ***Benchmark Datasets and Evaluation Metrics for Attacks****: Empirical studies on SL attacks are predominantly grounded in a narrow set of vision benchmarks. The most extensively used datasets are MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, ImageNet subsets, and CelebA, while other datasets such as AT&T, CINIC-10, Fruits-360, Dogs vs. Cats, SVHN, Criteo, PTB-XL, and GTSRB appear only rarely across the literature. Then, attack effectiveness is typically measured using reconstruction fidelity metrics (MSE, PSNR, SSIM), label and attribute inference accuracy, ASR, CDA, and class-wise recall degradation. However, variations in datasets, metric usage, cut-layer configurations, and adversarial assumptions across studies limit the consistency and comparability of results, making fair side-by-side comparison difficult and thus challenging RQ3.*

Hybrid-SL paradigms, such as SFL, exhibit an attack surface that combines threats inherited from FL with those unique to the SL. Inherited FL-style attacks include gradient-based label inference, poisoning-based attacks such as label flipping, backdoor attacks, adversarial attacks, and functionality stealing. These threats transfer directly to Hybrid-SL whenever gradient exchange is present. In contrast, SL-specific attacks arise from the exposure of smashed activations at the cut layer, enabling feature reconstruction attacks, FSHA, activation-level model inversion, GAN-based reconstruction, smashed data-based label inference, and property inference attacks. Because these attacks exploit intermediate activations rather than gradients, they do not appear in FL.

4.2 Attack Constraints

Attack constraints capture the technical and operational requirements for adversaries in SL. We outline those revealed by our findings below:

Dataset Constraints. Dataset constraints influence the vulnerability landscape of SL systems. One fundamental constraint is the adversary's ability to access domain-similar datasets, which substantially enhances the feasibility of various attack vectors [54]. Studies have shown that an exact match between the adversary's dataset and the target data is not necessary; it is sufficient for

the datasets to share similar distributional characteristics to mount effective attacks [24, 27, 78, 85]. Furthermore, access to publicly available datasets enables the development of more sophisticated threats. For instance, adversaries can train surrogate models to reconstruct private data [82, 83] or employ shadow models to conduct backdoor insertion and poisoning attacks [29, 80, 81].

Knowledge-Based Constraints. The assumption that the server possesses complete knowledge of the learning task significantly amplifies the effectiveness of adversarial strategies. Under this constraint, adversaries are able to design task-specific queries that exploit the model's learning objective, thereby increasing the risk of privacy breaches [54, 83]. For instance, in a setting where the learning task involves medical imaging, adversaries can synthesize patient-specific scans to infer sensitive attributes. Additional assumptions, such as the server and client sharing the same optimizer, further facilitate attack success by enabling more accurate reconstruction of training data or inference of model parameters [83]. Some works explore scenarios where the adversary operates with only partial information about the learning objective [14].

Architecture Exposure Constraints. Architecture exposure constraints play a crucial role in determining the vulnerability of SL systems to various attacks. Exposure to the client-side architecture significantly enhances the efficacy of model manipulation [24, 27], inversion [10], and feature reconstruction attacks [85]. Building on this constraint, other threat models assume adversaries have full knowledge of the client's subnetwork and the underlying data distribution [67]. Thus, it strengthens the potential for backdoor attacks, enabling adversaries to manipulate the model's internal structure and implement covert functionalities while evading detection.

Label and Classification Constraints. Knowledge of the number of discrete labels in the dataset allows adversaries to refine their predictions and narrow the search space for classification inference [10]. In some settings, adversaries are further assumed to employ binary classifiers to predict labels from observed model activations, a technique commonly leveraged in membership inference attacks to determine the presence of specific samples in the training set [42]. Additional assumptions, such as requiring only a single labeled sample per class or having access to all participant labels, further strengthen the adversary's capability to infer broader label information from model activations [4, 45].

Observation. ***Architecture:*** *SL system's architecture strongly influences its privacy risks. Most studies focus on conventional VanSL, which uses a single cut and is especially prone to feature hijacking [78, 85] and label inference [33,42]. In contrast, USL, Hybrid-SL, and SFL use multi-hop client–server interactions that create different attack surfaces, particularly for label leakage and interception of intermediate representations. NLSL and MHSL are notably underexplored. NLSL boosts privacy via client-side label processing, while MHSL enables chained computation across parties. These relate directly to RQ1, which aims to classify adversarial threats across SL variants.*

Open Problem. ***Variant-Specific Threat Models and Defenses:*** *SL research lacks threat models and defenses tailored to variants beyond VanSL. For example, MHSL introduces unique concerns such as multi-point leakage aggregation, where an adversary could compromise multiple cut layers across hops. Similarly, NLSL's emphasis on local label preservation is vulnerable to activation inference or gradient-linkage attacks. Existing defense mechanisms, largely designed for VanSL, fail to generalize to these topologies. Also, trust assumptions and communication protocols are rarely modeled, creating a major blind spot. In the absence of clear formalizations of attack goals and defense feasibility, security guarantees remain speculative.*

4.3 Attack Effectiveness

Domain-Independent Attacks. These attacks are dataset- and model-agnostic, and they exploit fundamental weaknesses in the SL process rather than domain-specific patterns, allowing them to generalize across multiple settings. For instance, [4, 10, 33, 42, 45, 54, 78, 83] attacks have been shown to reconstruct input data and inference labels across diverse learning environments, demonstrating that their effectiveness is not constrained to a particular application.

Stealth. In traditional security frameworks, attacks are often identified by monitoring anomalies. However, many SL attacks [10, 20, 24, 29, 45, 54, 63, 77, 78, 81, 83–85] evade detection by ensuring that they do not introduce observable deviations in gradient updates or model performance. For instance, distance correlation minimization is leveraged in [10, 54, 78] to launch stealthy attacks, where gradient updates are subtly manipulated while preserving the model behavior.

Defeating Differential Privacy (DP). DP [9] is a leading privacy-preserving method in ML, which adds controlled noise to training updates to ensure that the presence or absence of any single data point has little effect on the model's output (see next section for details). However, recent studies [4, 21, 27, 42, 45, 83, 84] have demonstrated that DP alone is insufficient to defend against advanced SL attacks. The main limitation of DP in SL is its focus on protecting individual data points rather than preventing large-scale reconstruction. Although DP reduces the sensitivity of activations and gradients, adversaries can still exploit their structural and statistical properties to recover private information at an aggregated level. Moreover, adversaries employing distance correlation minimization techniques [10, 54, 78] circumvent DP protections. By iteratively optimizing gradient representations, adversaries can reconstruct highly accurate approximations of the original input data, reducing the effectiveness of DP.

SL Variants. The effectiveness of attacks in SL is not confined to a single variant but extends across all architectures [14, 83–85], demonstrating a fundamental vulnerability. Since all SL variants inherently rely on the exchange of intermediate activations or gradients between clients and servers, adversaries can exploit this to infer sensitive information, manipulate learning dynamics, or introduce adversarial perturbations. The structural modifications between VanSL, USL, and Hybrid-SL do not provide sufficient protection, as the core vulnerability (gradient leakage and model influence) remains consistent across variants.

Observation. ***Reliance on Strong Adversarial Capabilities:*** *Many prominent SL attack studies assume strong adversarial capabilities, such as full or partial model access [54, 82] or access to auxiliary datasets aligned with the training distribution [29,80,83]. While these reveal theoretical risks, they often diverge from real-world settings with limited adversarial access. To address RQ3, we differentiate strong vs. constrained assumptions, supporting more realistic threat models and the development of practical, context-aware defenses.*

Open Problem. ***Evaluating Practical Threats Under Realistic Conditions:*** *In response to RQ3, future research should prioritize the development and evaluation of attacks under more constrained and realistic conditions. This includes black-box settings, limited or mismatched auxiliary data, bounded query or compute budgets, and scenarios involving partial or probabilistic knowledge of the target model. Crucially, quantifying how attack effectiveness degrades under practical constraints is vital for accurate and actionable threat models for real-world SL deployments.*

5 Defense For Split Learning

In Section 4, we explored how adversaries exploit SL vulnerabilities. In response, various defense strategies have emerged. In this section, we systematically analyze defense mechanisms based on *defense strategies* (Section 5.1), *constraints* (Section 5.2), and *effectiveness* (Section 5.3).

5.1 Defense Strategies

In this section, we present a taxonomy and overview of defense strategies. Their distribution in the surveyed literature is shown in Fig. 5, and the taxonomy in Fig. 6. We note here that we classify defenses first into protection and detection. In SL, the boundary between privacy and security attacks is blurred: a single action, such as gradient manipulation, may both leak sensitive data and undermine model integrity. Our practitioner-oriented hierarchy clarifies how to distinguish between proactive defenses and reactive detection, emphasizing the interconnected vulnerabilities present at the cut layer.

Protection Mechanisms Protection mechanisms aim to secure parties preemptively before attacks occur. These approaches generally limit the information an adversary can extract, often assuming a *semi-honest* threat model.

Data Perturbation. In this approach, clients intentionally *modify* (e.g., add noise to) data or intermediate representations, aiming to obscure sensitive information. For example, DP gained prominence for its formal, provable privacy guarantees. By calibrating noise according to (ε, δ)-DP, data contributors can formally bound the risk. The privacy guarantee is established through its definition: $\Pr[\mathcal{M}(\mathcal{X}) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(\mathcal{X}') \in S] + \delta$ where $\mathcal{M}$ is a randomized mechanism that takes a dataset $\mathcal{X}$ as input and $\mathcal{M}$ satisfies (ε, δ)-DP for all

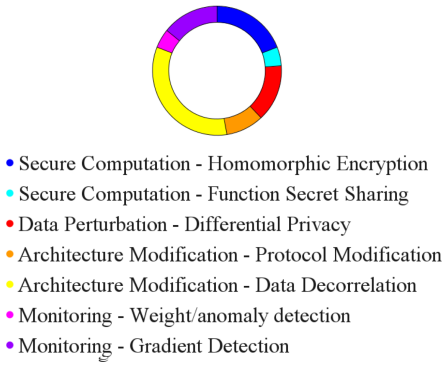

Fig. 5. Distribution of papers from the literature on defense mechanisms

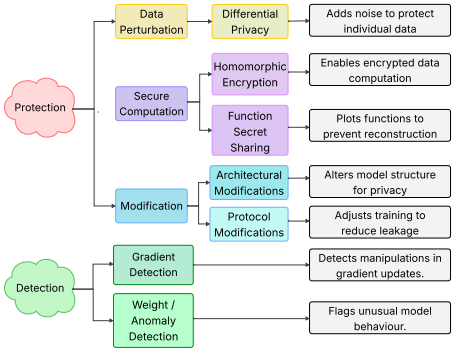

Fig. 6. Taxonomy of Defense Strategies

neighboring datasets $\mathcal{X}$ and $\mathcal{X}'$ that differ at most one entry. Here, ε defines the privacy budget and δ is a small probability of violating ε-DP. DP is often achieved by adding noise (e.g., Laplace, Gaussian, uniform) to data or query results, obscuring the impact of any individual. In the context of SL, DP is often applied to perturb the activations exchanged between the client and server, preventing sensitive information leakage through intermediate representations. Nonetheless, DP methods often entail a trade-off between privacy and model accuracy. Key challenges in these schemes include:

– *Accuracy vs. Noise*: Elevated noise degrades model performance.
– *Scalability*: Tuning DP parameters for high-dimensional data or large-scale models is non-trivial.

Several papers [21, 52, 71] investigate how DP needs to be applied in terms of both achieving privacy and accuracy in various SL setups.

Secure Computation. Cryptographic methods ensure security by limiting raw data access, each preserving specific properties. *Homomorphic encryption (HE)* enables computations on encrypted data. Formally, if E represents the encryption function, D the decryption function, and $\circ$ a supported operation (such as addition) under HE, then an HE scheme satisfies: $D(E(x) \circ E(y)) = x \circ y$ where x and y are plaintext values, and $E(x)$ and $E(y)$ are their corresponding ciphertexts. Each encryption scheme provides different functionalities, and in the context of SL, fully homomorphic encryption is preferred for enabling arbitrary computations on encrypted data without decryption.

On the other hand, *function secret sharing (FSS)* is a cryptographic technique that splits a function into n shares, distributed across parties, such that the original function is revealed only when the shares are combined. Below, we present the formal definition of FSS as outlined in [5].

For $n \in \mathbb{N}$, an n-party FSS scheme with respect to a share output decoder $\mathrm{DEC} = (S_1, \ldots, S_n, R, \mathrm{Dec})$ and a function class $\mathcal{F}$ is a pair of probabilistic polynomial-time algorithms $(\mathrm{Gen}, \mathrm{Eval})$ defined as:

- **Key Generation:** The algorithm $\text{Gen}(1^\lambda, f)$ with a security parameter λ and a function description $f \in \mathcal{F}$, outputs n secret keys $(k_1, \ldots, k_n)$.
- **Evaluation:** Each party i runs $\text{Eval}(i, k_i, x)$, with a party index i, the corresponding key k_i, and an input x. It outputs a value $y_i \in S_i$, representing the party's share of $f(x)$.

For instance, with two non-colluding semi-honest servers, neither can reconstruct the training function alone, yet together they can compute the result as $f(x) = f_1(x) \oplus f_2(x)$ where f_1, f_2 are computationally indistinguishable. The primary goal is not simply to guarantee data privacy-an aim already addressed by well-established cryptographic schemes—but rather to optimize the application of these strategies for efficiency and practicality, particularly in resource-constrained environments. Khann et al. [34] explore a two-server FSS approach, examining how different secure computation techniques can be adapted for SL.

HE is generally applied to specific workflows within SL, such as encrypting activations and gradients, allowing the server to compute on confidential data without accessing raw information. To mitigate the high computational overhead, hybrid approaches are also often employed where non-confidential parts of the model are processed in plaintext. Consequently, HE is primarily used to securely aggregate client data or model updates. Other works [31,38,53,56] which integrate HE into SL, combine cryptographic methods to achieve secure split learning. Though these methods preserve accuracy and security, their computational cost can be impractical in resource-constrained settings.

Modifications. Defense strategies in this section modify the model architecture or the protocol to limit information leakage, which we detail below:

a. Architectural Modifications: Architectural modifications introduce structural changes to SL to prevent reconstruction attacks. Abuadbba et al. [1] investigate the effects of shrinking the cut layer on the effectiveness of FSHA. Pham et al. [57] introduce a binarized neural network combined with DP to analyze its impact on security and efficiency. Another approach involves *specialized activation functions*, such as randomized ReLu [49], which introduces randomness to obscure patterns and reduce information leakage. Additionally, mechanisms designed to de-correlate transmitted data have been proposed in [39,49,66,70,75], mitigating information leakage while preserving utility.

Observation. ***Architectural Changes:*** *Architectural modifications can limit exposure and hinder attacks, but these measures alone rarely ensure provable privacy, serving best as complements to stronger defenses. This observation addresses RQ5, which concerns the suitability and constraints of defense strategies for specific tasks.Architectural changes are thus well-suited for low-resource settings or serve as lightweight complements to stronger defenses.*

b. Protocol Modifications: Refining the protocol that governs data exchange in SL can substantially strengthen privacy. A notable example is SL without local weight sharing (P-SL) [61], which prevents adversaries from leveraging model inversion to reconstruct client data. Instead of sharing model parameters,

this approach enables collaborative training where the gradient is computed as:

$$\nabla L(\text{outputs}, \text{labels}) = \nabla_{u_i,w} L\left(g_w([z_i, z^{\text{cache}}]), [y_i^{\text{train}}, y^{\text{cache}}]\right)$$

where $z_i = f_u(x_i)$ is the smashed data, and z_{cache} the previously cached smashed data, effectively mitigating leakage from individual updates.

An alternative strategy is to *invert the SL setup*, wherein clients hold the labels while the server processes encrypted features [31]. This reduces client computation without compromising utility. Other examples include [41, 61], which explore communication and data-sharing regulations. Such protocol modifications ensure that client-side computations remain efficient while reducing the effectiveness of FSHA and reconstruction attacks [41, 70]. However, both architecture and protocol level modifications introduce a computational burden on the server, which must be managed to maintain scalability. While empirically effective, these methods may reduce accuracy and often lack formal security guarantees, making them vulnerable to slight variations in attack strategies. Key considerations include:

- *Model Accuracy vs. Privacy*: Structural modifications may compromise utility.
- *Robustness Under Diverse Attacks*: Without formal guarantees, even minor changes in attack tactics might bypass these defenses.

Detection Mechanisms Detection mechanisms focus on *monitoring* the training process to identify malicious activity *during or after* its occurrence, thereby enabling timely responses such as halting or rolling back updates. These methods are particularly relevant for scenarios involving overtly malicious adversaries who might engage in label-flipping, backdoor insertion, or model hijacking. Two primary detection approaches are employed:

Gradient/Update Monitoring. This method involves analyzing the gradients or updates exchanged between clients and servers to detect abnormal variations. Outlier detection techniques can highlight significant deviations that may signal an ongoing attack. Several studies have explored gradient anomaly detection to identify adversarial activity in SL [11, 12, 18]. Considering the approach in [18], the detection score DS_n is computed as $DS_n = Sig(\lambda_{ds}(\hat{G}_n \cdot \hat{E}_n \cdot \hat{V}_n - \alpha))$ where $\hat{G}_n$, $\hat{E}_n$, and $\hat{V}_n$ represent adjusted gradient similarity gap, approximation error, and class-wise gradient overlap, respectively. The sigmoid function Sig normalizes the detection score to $[0, 1]$, enabling detection of adversarial gradient manipulations via deviations in gradient behavior.

Weight Anomaly Detection. Some frameworks monitor weight changes to detect anomalies and adversarial tampering. For example, [63] analyzes weight distributions to spot backdoor attacks. An example anomaly detection metric is *Rotational Distance* [63], which measures directional changes in parameter space across training steps: $\theta(t) = \arccos\left(\frac{\mathbf{B}_t \cdot \mathbf{B}_{t-1}}{\|\mathbf{B}_t\| \|\mathbf{B}_{t-1}\|}\right)$ where $\mathbf{B}_t$ and $\mathbf{B}_{t-1}$ are backbone parameters at consecutive training steps. The angular displacement $\theta(t)$ measures the directional shift in model updates. However, backdoor attacks introduce abnormal directional changes in weight updates, causing $\theta(t)$ to deviate

from normal training patterns. By averaging pairwise differences in $\theta(t)$ across clients, anomalies can be detected and flagged as potential threats.

Monitoring techniques have advanced to strengthen security while preserving integrity. Rieger et al. [63] propose periodically auditing trained models to verify participant honesty, emphasizing the risk of mislabeling honest users as malicious while maintaining robust training with minimal false positives.

Observation. ***Prominence of Studied Defenses:*** *The literature shows DP as the most extensively studied approach for enhancing confidentiality in SL, followed by HE. In contrast, detection research focuses primarily on gradient analysis. Highlighting these commonly used techniques directly informs RQ4.*

5.2 Defense Constraints

Threat Model Constraints. Alongside the attacks discussed in Section 4.1, defense strategies differ in how they address adversaries that may only observe and infer (semi-honest) versus those that actively alter the environment (malicious). For semi-honest scenarios, cryptographic methods [31,34,37,53,56], data perturbation [59,69,71], and architecture-based approaches [1,41,49,57,66] generally suffice, as they aim to mitigate reconstruction attacks. In malicious settings, verification-based detection strategies [11,13,18,63] are often necessary to detect and prevent adversarial behaviors such as poisoning or backdoor attacks.

Multi-Party Verification Constraints. When multiple clients or servers are involved, defenses may need enhanced validation mechanisms to differentiate between benign and malicious updates [11, 13, 18]. Some methods [39, 63] rely on cross-comparisons among clients or a trusted third party, which increases communication rounds and latency. In large-scale or bandwidth-limited environments, frequent verification may be impractical. Leveraging powerful nodes can mitigate some of these issues, but introduces additional trust assumptions.

Architectural Constraints. Certain defenses [1, 57] depend on the model's partitioning and the number of layers retained on the client side [57] or specialized activation functions [49]. However, this can reduce flexibility or performance on complex tasks. Architectural considerations become increasingly crucial for a generalized approach that minimizes reliance on specific empirical setups.

Computational and Operational Constraints. Practical SL deployments, especially on edge or mobile devices, restrict the use of encryption or gradient checks. Heavy cryptographic methods are often impractical, prompting a focus on improving computational efficiency [31, 37, 53]. Defenses assuming constant client availability can be disrupted by dropouts or delays [63]. These challenges underscore the need for robust strategies for network and resource variability.

Open Problem. ***Broadening Threat Models:*** Future defenses should be evaluated under a threat spectrum that extends beyond the semi-honest assumption, including fully malicious, colluding, or adaptive adversaries. Considering stronger adversarial models, such as Byzantine, covert, or adaptive frameworks, is essential for realistic deployment scenarios.

5.3 Defense Effectiveness

Defense effectiveness in SL is evaluated based on several key factors that we detail below. However, we note here that quantitative comparison of defenses remains challenging due to the lack of standardized benchmarks. Most existing works do not publish open-source implementations, and the diversity in attack objectives and defense settings makes fair side-by-side comparison difficult. Consequently, assessing the trade-off between defense effectiveness and utility is often limited to individual study settings rather than a unified framework.

Observation. ***Benchmark Datasets and Evaluation Metrics for Defense***: Defense studies' evaluations mirror the dataset usage of attack literature, primarily relying on image (MNIST, CIFAR-10) datasets to demonstrate mitigation capabilities, while occasionally incorporating medical time-series (e.g., PTB-XL). Quantitatively, defense effectiveness is assessed through a multi-dimensional trade-off. Privacy gains are measured by reduced adversarial success (e.g., lower SSIM/PSNR or decreased inference and backdoor attack accuracy), whereas utility is measured by the accuracy drop relative to a baseline (non-private solution). Efficiency is measured in terms of communication and computational overhead, but no standardized evaluation framework exists. Because defenses are evaluated under diverse threat models and constraints, the lack of baselines prevents direct quantitative comparison, including the practical distinctions between cryptographic and perturbation-based methods.

Observation. ***Confidentiality & Integrity:*** *Most SL defenses focus on two main goals: confidentiality (protecting data, labels, or attributes) and integrity (preventing model manipulation). Availability, however, remains largely underexplored. This distinction informs RQ6—how defenses prioritize confidentiality vs. integrity—and aligns with the taxonomy for RQ4. Notably, detection strategies, which are key for identifying active threats like poisoning or backdoors (though limited against passive leakage), also fall within this classification.*

Phase of Operation (training vs. inference) Data perturbation, architectural and protocol modifications, and gradient monitoring are applied during the training phase. In contrast, cryptographic methods secure computations during both training and inference. Detection mechanisms are varied; while gradient-based checks are training-specific, others that monitor model weights or flag unusual behavior can be active during training or inference.

Attack Coverage and Defense Alignment We summarize the mapping of SL attacks and corresponding defense mechanisms in Table 1. Each defense addresses specific adversarial threats in SL. *Data perturbation defenses* [1,59,69,71], including DP and feature decorrelation, are effective against *label inference and reconstruction attacks*, making it more difficult for adversaries to extract information from gradients or intermediate representations. However, these ap-

Table 1. Mapping of SL attacks to corresponding defense mechanisms. The "Defense Shown Ineffective" column lists defenses that fail against the corresponding attack.

Attack	Defense Mechanism	Defenses Shown Ineffective
Data Reconstruction	Secure Computation [31, 34], Architectural Modifications [57, 74], Protocol Modification [61], Anomaly detection (gradient) [18]	Correlation-aware reconstruction [10, 54, 82, 85], autoencoder inversion [83], adaptive denoising [78, 83], DP [78]
Label Inference	DP [59], HE [31, 38], Architectural modifications [75]	Gradient compression [45, 84], noise obfuscation [27, 77, 84]
Backdoor Attack	Anomaly detection(gradient, weight) [18, 63]	Gradient compression [4]
Adversarial Attack	Data perturbation [69], Architectural modifications [49]	Adaptive perturbations [24]
Poisoning Attack	Anomaly detection (gradient, weight) [13, 63], Architectural Modifications [39]	Intraclass-Distance Inflated Loss [40], Byzantine-Resilient Aggregation Methods [76]

proaches often introduce a trade-off in accuracy. *Secure computation techniques* [31, 34, 38, 53, 56], such as HE and FSS, offer robust privacy guarantees.

Observation. ***Privacy–Utility–Overhead Trade-off:*** *SL defenses show a clear trade-off between privacy, utility, and overhead. Methods range from lightweight approaches with weaker guarantees to heavy cryptographic schemes with high costs. Although DP offers a tunable middle ground, balancing it remains application-specific. This relates to RQ4 by situating cryptographic techniques among SL defenses, to RQ5 on suitability and optimization, and to RQ6 on what constitutes a balanced,* successful *defense.*

Architectural modifications [1, 39, 41, 49, 57, 61, 66, 70, 74, 75] mostly mitigate *FSHA and reconstruction attacks*, but their effectiveness depends on the complexity of the model. Detection mechanisms, on the other hand, are crucial for identifying adversarial actions during training. *Gradient anomaly detection* [11, 13, 18] is effective against *poisoning and hijacking attacks* while *model weight anomaly detection* [63] is particularly useful against *backdoor attacks* as it can identify hidden manipulations within the model weights. The effectiveness of these mechanisms depends on accurate profiling of training, but subtle adversarial manipulations can evade detection.

Observation. ***Hybrid Defenses:*** *While combining defenses may seem promising, studies show that interactions-such as applying DP over encrypted data or adding noise under architectural constraints-can raise overhead, weaken guarantees, or introduce new vulnerabilities. These findings, central to RQ5, emphasize that effective composition requires careful joint analysis.*

Open Problem. ***Advancing Holistic Monitoring Strategies:*** *Future work should emphasize proactive, multi-layered defenses for SL's broad attack surface. Combining anomaly detection with FL-inspired techniques such as secure aggregation for client verification [32] can strengthen detection without adding significant overhead. Defenses against side-channel threats (e.g., [8]) and secure inference can enhance robustness against covert attacks. Thus, achieving holistic SL security may require hybrid approaches, e.g., combining secure computation for confidentiality with monitoring for integrity. A key challenge is integrating monitoring with other defenses with low overhead.*

Client-Server Setup and Deployment Scenarios The client-server setup influences the feasibility and effectiveness of defenses. In *single-client SL (SCSL)*, cryptographic defenses such as HE and FSS are more feasible, as cryptographic overhead is limited to a single party. Architectural modifications are also easier to apply, as there is no need for cross-client consistency. In *multi-client SL (MCSL)*, however, *client-side gradient monitoring* gains importance due to the risk of adversaries hiding among honest clients. *Data perturbation techniques*, such as DP, require careful tuning in MCSL due to data heterogeneity, and alone offer limited defense as discussed in Section 4.3. *Secure computation methods* face scalability challenges, as the overhead increases with an increasing number of clients. In MCSL, *architectural modifications* such as split-layer depth optimization and secure aggregation become more practical, as computational burdens are distributed across multiple parties. Lastly, communication latency and encryption overhead remain key concerns for cryptographic techniques.

Open Problem. ***Instance Encoding:*** *Intermediate representations in SL (smashed data) can leak sensitive information even without full input reconstruction, a concern known as the instance encoding [6, 47]. Carlini et al. [6] reframed PPML as instance encoding, analyzing its statistical basis, comparing methods, and providing bounds and empirical results under specific attacks. Given the alignment in goals and definitions, this concept provides valuable insights for improving the security of SL.*

Model and Dataset Considerations Defense effectiveness also depends on the complexity of the model architecture and the dataset's properties. In *lightweight models* such as convolutional neural networks, *DP and gradient perturbation* are effective [59, 69], as small distortions in features do not drastically impact performance. Cryptographic methods such as HE remain feasible in small-scale models [31, 36]. In *deep architectures* such as transformers, *secure computation techniques* struggle due to their high computational complexity, while *architectural modifications* [1, 49, 57] become more critical to mitigate FSHA. *Detection-based methods*, such as gradient anomaly analysis, strongly depend on model architecture and training setup, since gradient information is highly influenced by these factors. For example, [11] notes that larger gradients face the curse of dimensionality, as these methods rely on distance and correlation analysis.

6 Open Research Directions

Building on our observations, we outline the key research directions below:

- **Information Bottleneck-Based Cut Layer Design:** To mitigate semantic leakage through smashed data, a promising direction is to incorporate the Information Bottleneck (IB) principle into the architecture design of SL. By formulating the cut-layer transformation $f_c : x \to z$ as an optimization problem that minimizes $I(x, z)$ while preserving $I(z, y)$, where y is the label, it becomes feasible to enforce representational compression. Leveraging techniques like deterministic approximations (e.g., dropout as noise injectors) could lead to empirically effective and theoretically grounded SL. These architectures could also be evaluated using entropy-aware metrics to compare leakage resilience across different deployment scenarios.
- **Topology-Aware and Trust-Adaptive Security Frameworks:** A promising direction is to develop topology-aware attack taxonomies and trust-adaptive defense frameworks. This involves modeling SL variants (e.g., NLSL, MHSL, Hybrid-SL) as a graph where nodes represent computational agents and edges represent data/gradient flows. Attack surfaces could then be systematically analyzed using graph traversal metrics (e.g., path entropy, betweenness centrality) to identify high-risk cut points. Defense strategies can be made variant-specific by integrating cryptographic techniques (e.g., secure relay [44] in MHSL) and adaptive noise mechanisms like context-sensitive DP in FL [7, 19, 68]. These structured, topology-driven strategies will provide scalable and secure SL systems for real-world deployments.
- **Toward a Layered Adversarial Evaluation:** Future research should aim to develop a progressive adversarial framework that evaluates defenses across a spectrum of adversarial strengths, ranging from semi-honest actors to fully malicious colluding parties. SL systems are especially vulnerable to adversaries capable of poisoning gradients or manipulating activations. We advocate for hybrid evaluation protocols that integrate empirical testing with theoretical analysis to assess robustness. Furthermore, adopting formal models from cryptographic literature (e.g., Byzantine threat models [15], covert adversaries [3], or adaptive adversaries [2]) can help bridge the current gap between theoretical security guarantees and practical adversarial capabilities.
- **Verifiable Training and Inference:** Verifiability of training and inference in SL, alongside privacy, is another crucial objective. Zero-knowledge (ZK)-proofs enable parties to prove correct computation without leaking unauthorized information, offering a promising means to verify the correctness of training. This capability is vital where malicious participants alter training data, introduce backdoors, or poison the training process. Peng et al. [55] survey ZK-proofs, categorizing them into verifiable training, testing, and inference, and propose a framework for their use in ML, which can be beneficial for SL settings. Yet, efficiency would constitute a crucial concern.
- **Exploring Hardware-Based Security Solutions:** Hardware-based solutions, such as trusted execution environments (e.g., [28]), are under-explored and show promise for enhancing security and privacy in SL.

– **Resilience to Distributed System Faults** Another key direction is ensuring training remains robust despite common distributed challenges like client dropouts, delays, and communication errors. Strategies should focus on maintaining training integrity even when parts of the system fail.

7 Related Work

Several SoK papers have examined PPML paradigms. Podschwadt et al. [62] systematized privacy-preserving deep learning using HE, highlighting its computational challenges and practical limitations. Ng and Chow [51] conducted an extensive SoK examining cryptographic approaches. Mansouri et al. [48] analyzed secure aggregation techniques for FL, categorizing encryption-based and MPC-based approaches. We remind here that FL and SL both avoid raw data sharing but differ in partitioning: FL shares gradients from local models, while SL exchanges cut-layer activations. Thus, defenses like secure aggregation [16] protect FL but do not directly mitigate SL's unique cut-layer vulnerabilities.

In examining SL-specific concerns, Pham and Chilamkurti [58] surveyed data leakage threats, identifying gradient leakage, label inference, and feature reconstruction attacks. Hu et al. [26] conducted a review and experimental evaluation of SL, highlighting the variability in SL paradigms concerning cut-layer selection, model aggregation, and label sharing. In parallel with our work, Khan et al. [35] extend the systematization of SL attacks. While these works provide valuable systematic insights, our SoK, to the best of our knowledge, is the first to introduce a formal taxonomy that jointly categorizes both attacks and defenses in SL while analyzing their effectiveness and limitations.

8 Conclusion

We systematically explored the security and privacy landscape of Split Learning (SL) by presenting a novel taxonomy of attack strategies and defense mechanisms. Employing the key observations and takeaways presented in this paper, future research can enhance the resilience of SL, making it a viable solution for secure collaborative learning.

Acknowledgements

We acknowledge TÜBİTAK (the Scientific and Technological Research Council of Türkiye) project 124N941. The authors utilized ChatGPT-4o for shortening sentences and correcting grammatical errors to enhance readability.

References

1. Abuadbba, S., Kim, K., Kim, M., Thapa, C., Camtepe, S.A., Gao, Y., Kim, H., Nepal, S.: Can we use split learning on 1d cnn models for privacy preserving training. In: ASIACCS (2020)

2. Arora, R., Dekel, O., Tewari, A.: Online bandit learning against an adaptive adversary: from regret to policy regret. arXiv:1206.6400 (2012)
3. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. Journal of Cryptology (2010)
4. Bai, Y., Chen, Y., Zhang, H., Xu, W., Weng, H., Goodman, D.: {VILLAIN}: Backdoor attacks against vertical split learning. In: USENIX Security (2023)
5. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: EUROCRYPT (2015)
6. Carlini, N., Deng, S., Garg, S., Jha, S., Mahloujifar, S., Mahmoody, M., Song, S., Thakurta, A., Tramèr, F.: Is private learning possible with instance encoding? In: IEEE S&P (2021)
7. Chen, Z., Zheng, H., Liu, G.: Awdp-fl: An adaptive differential privacy federated learning framework. Electronics (2024)
8. Debenedetti, E., Severi, G., Carlini, N., Choquette-Choo, C.A., Jagielski, M., Nasr, M., Wallace, E., Tramèr, F.: Privacy side channels in machine learning systems. In: USENIX Security (2024)
9. Dwork, C.: Differential privacy. In: ICALP (2006)
10. Erdoğan, E., Küpçü, A., Çiçek, A.E.: Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In: ACM WPES (2022)
11. Erdogan, E., Teksen, U., Celiktenyildiz, M.S., Kupcu, A., Cicek, A.E.: Splitout: Out-of-the-box training-hijacking detection in split learning via outlier detection. In: CANS (2024)
12. Erdoğan, E., Küpçü, A., Çiçek, A.E.: Splitguard: Detecting and mitigating training-hijacking attacks in split learning. In: ACM WPES (2022)
13. Erdoğan, E., Tekşen, U., Çeliktenyıldız, M.S., Küpçü, A., Çiçek, A.E.: Defense mechanisms against training-hijacking attacks in split learning. IEEE TKDE (2023)
14. Fan, M., Chen, C., Wang, C., Zhou, W., Huang, J.: On the robustness of split learning against adversarial attacks. In: ECAI (2023)
15. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to {Byzantine-Robust} federated learning. In: USENIX Security (2020)
16. Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Möllering, H., Nguyen, T.D., Rieger, P., Sadeghi, A.R., Schneider, T., Yalame, H., et al.: Safelearn: Secure aggregation for private federated learning. In: IEEE SPW (2021)
17. Fu, C., Zhang, J., Zhu, T., Zhou, W., Yu, P.S.: Label inference attacks against vertical federated learning. In: USENIX Security (2022)
18. Fu, J., Ma, X., Zhu, B.B., Hu, P., Zhao, R., Jia, Y., Xu, P., Jin, H., Zhang, D.: Focusing on pinocchio's nose: A gradients scrutinizer to thwart split-learning hijacking attacks using intrinsic attributes. In: NDSS (2023)
19. Fu, J., Chen, Z., Han, X.: Adap dp-fl: Differentially private federated learning with adaptive noise (2022)
20. Gajbhiye, S., Singh, P., Gupta, S.: Data poisoning attack by label flipping on splitfed learning. In: RTIP2R (2022)
21. Gawron, G., Stubbings, P.: Feature space hijacking attacks against differentially private split learning. arXiv:2201.04018 (2022)
22. Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., Sriperumbudur, B.K.: Optimal kernel choice for large-scale two-sample tests. NIPS (2012)
23. Gupta, O., Raskar, R.: Distributed learning of deep neural network over multiple agents (2018)

24. He, Y., Hu, C., Pu, Y., Chen, J., Li, X.: Advusl: Targeted adversarial attack against u-shaped split learning. In: IEEE MASS (2024)
25. He, Z., Zhang, T., Lee, R.B.: Model inversion attacks against collaborative inference. In: ACSAC (2019)
26. Hu, Z., Zhou, T., Wu, B., Chen, C., Wang, Y.: A review and experimental evaluation on split learning. Future Internet (2025)
27. Huang, H., Li, X., He, W.: Pixel-wise reconstruction of private data in split federated learning. In: ICICS (2023)
28. Huang, W., Wang, Y., Cheng, A., Zhou, A., Yu, C., Wang, L.: A fast, performant, secure distributed training framework for llm. In: ICASSP (2024)
29. Ismail, A.T.Z., Shukla, R.M.: Analyzing the vulnerabilities in splitfed learning: Assessing the robustness against data poisoning attacks. arXiv:2307.03197 (2023)
30. Joshi, P., Thapa, C., Camtepe, S., Hasanuzzaman, M., Scully, T., Afli, H.: Performance and information leakage in splitfed learning and multi-head split learning in healthcare data and beyond. Methods and Protocols (2022)
31. Kanpak, H.I., Shabbir, A., Genç, E., Küpçü, A., Sav, S.: CURE: Privacy-preserving split learning done right. arXiv:2407.08977 (2024)
32. Karakoç, F., Küpçü, A., Önen, M.: Fault tolerant and malicious secure federated learning. In: CANS (2024)
33. Kariyappa, S., Qureshi, M.K.: Exploit: Extracting private labels in split learning. In: SaTML. IEEE (2023)
34. Khan, T., Budzys, M., Michalas, A.: Make split, not hijack: Preventing feature-space hijacking attacks in split learning. In: SACMAT (2024)
35. Khan, T., Michalas, A.: Oops!... they stole it again: Attacks on split learning. arXiv:2508.10598 (2025)
36. Khan, T., Nguyen, K., Michalas, A.: A more secure split: Enhancing the security of privacy-preserving split learning. In: AsiaCCS (2023)
37. Khan, T., Nguyen, K., Michalas, A.: Split ways: Privacy-preserving training of encrypted data using split learning. In: arXiv:2301.08778 (2023)
38. Khan, T., Nguyen, K., Michalas, A., Bakas, A.: Love or hate? share or split? privacy-preserving training using split learning and homomorphic encryption (2023)
39. Khowaja, S.A., Lee, I.H., Dev, K., Jarwar, M.A., Qureshi, N.M.F.: Get your foes fooled: Proximal gradient split learning for defense against model inversion attacks on iomt data. IEEE TNSE (2024)
40. Kohankhaki, M., Ayad, A., Barhoush, M., Schmeink, A.: Detecting data poisoning in split learning using intraclass-distance inflated loss. In: IEEE GC Wkshps (2023)
41. Li, J., Rakin, A.S., Chen, X., He, Z., Fan, D., Chakrabarti, C.: Ressfl: A resistance transfer framework for defending model inversion attack in split federated learning. In: CVPR (2022)
42. Li, O., Sun, J., Yang, X., Gao, W., Zhang, H., Xie, J., Smith, V., Wang, C.: Label leakage and protection in two-party split learning. arXiv:2102.08504 (2018)
43. Li, Z., Yan, C., Zhang, X., Gharibi, G., Yin, Z., Jiang, X., Malin, B.A.: Split learning for distributed collaborative training of deep learning models in health informatics. In: AMIA Annu. Symp. Proc (2024)
44. Lin, Z., Liu, H., Zhang, Y.J.A.: Relay-assisted cooperative federated learning. IEEE TWC (2022)
45. Liu, J., Lyu, X., Cui, Q., Tao, X.: Similarity-based label inference attack against training and inference of split learning. IEEE TIFS (2024)
46. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: PMLR (2015)

47. Maeng, K., Guo, C., Kariyappa, S., Suh, G.E.: Bounding the invertibility of privacy-preserving instance encoding using fisher information. NeurIPS (2023)
48. Mansouri, M., Önen, M., Jaballah, W.B., Conti, M.: Sok: Secure aggregation based on cryptographic schemes for federated learning. PoPETs (2023)
49. Mao, Y., Xin, Z., Li, Z., Hong, J., Yang, Q., Zhong, S.: Secure split learning against property inference, data reconstruction, and feature space hijacking attacks. In: ESORICS (2023)
50. Nasr, M., Shokri, R., Houmansadr, A.: Machine learning with membership privacy using adversarial regularization. In: ACM SIGSAC (2018)
51. Ng, L.K., Chow, S.S.: Sok: cryptographic neural-network computation. In: IEEE S&P (2023)
52. Ngoc Duy Pham, K.T.P., Chilamkurti, N.: Enhancing accuracy-privacy trade-off in differentially private split learning (2024)
53. Nguyen, K., Khan, T., Michalas, A.: Split without a leak: Reducing privacy leakage in split learning. In: SecureComm (2025)
54. Pasquini, D., Ateniese, G., Bernaschi, M.: Unleashing the tiger: Inference attacks on split learning. In: ACM CCS (2021)
55. Peng, Z., Wang, T., Zhao, C., Liao, G., Lin, Z., Liu, Y., Cao, B., Shi, L., Yang, Q., Zhang, S.: A survey of zero-knowledge proof based verifiable machine learning (2025)
56. Pereteanu, G.L., Alansary, A., Passerat-Palmbach, J.: Split he: Fast secure inference combining split learning and homomorphic encryption (2022)
57. Pham, N.D., Abuadbba, A., Gao, Y., Phan, T.K., Chilamkurti, N.: Binarizing split learning for data privacy enhancement and computation reduction (2022)
58. Pham, N.D., Chilamkurti, N.: Data leakage threats and protection in split learning: A survey. In: ICEA (2023)
59. Pham, N.D., Phan, K.T., Chilamkurti, N.: Enhancing accuracy-privacy trade-off in differentially private split learning. IEEE TIFS (2024)
60. Pham, N.D., Phan, T.K., Abuadbba, A., Gao, Y., Nguyen, D., Chilamkurti, N.: Split learning without local weight sharing to enhance client-side data privacy. arXiv:2212.00250 (2022)
61. Pham, N.D., Phan, T.K., Abuadbba, A., Gao, Y., Nguyen, V.D., Chilamkurti, N.: Split Learning without Local Weight Sharing To Enhance Client-side Data Privacy. IEEE TDSC (5555)
62. Podschwadt, R., Takabi, D., Hu, P.: Sok: Privacy-preserving deep learning with homomorphic encryption. arXiv:2112.12855 (2021)
63. Rieger, P., Pegoraro, A., Kumari, K., Abera, T., Knauer, J., Sadeghi, A.R.: Safesplit: A novel defense against client-side backdoor attacks in split learning. In: NDSS (2025)
64. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D (1992)
65. Ryan, M.D.: Cloud computing privacy concerns on our doorstep. Communications of the ACM (2011)
66. Singh, A., Chopra, A., Sharma, V., Garza, E., Zhang, E., Vepakomma, P., Raskar, R.: Disco: Dynamic and invariant sensitive channel obfuscation for deep neural networks (2021)
67. Tajalli, B., Ersoy, O., Picek, S.: On feasibility of server-side backdoor attacks on split learning. In: IEEE SPW (2023)
68. Talaei, M., Izadi, I.: Adaptive differential privacy in federated learning: A priority-based approach. arXiv:2401.02453 (2024)

69. Titcombe, T., Hall, A.J., Papadopoulos, P., Romanini, D.: Practical defences against model inversion attacks for split neural networks. In: ICLR Workshop on DPML (2021)
70. Turina, V., Zhang, Z., Esposito, F., Matta, I.: Federated or split? a performance and privacy analysis of hybrid split and federated learning architectures. In: CLOUD (2021)
71. Vepakomma, P., Balla, J., Raskar, R.: Privatemail: Supervised manifold learning of deep features with differential privacy for image retrieval (2021)
72. Vepakomma, P., Gupta, O., Dubey, A., Raskar, R.: Reducing leakage in distributed deep learning for sensitive health data. arXiv:1812.00564 (2019)
73. Vepakomma, P., Gupta, O., Swedish, T., Raskar, R.: Split learning for health: Distributed deep learning without sharing raw patient data (2018)
74. Vepakomma, P., Singh, A., Gupta, O., Raskar, R.: Nopeek: Information leakage reduction to share activations in distributed deep learning (2020)
75. Wan, X., Sun, J., Wang, S., Chen, L., Zheng, Z., Wu, F., Chen, G.: Pslf: Defending against label leakage in split learning. In: ACM CIKM (2023)
76. Wu, X., Yuan, H., Li, X., Ni, J., Lu, R.: Evaluating security and robustness for split federated learning against poisoning attacks. IEEE TIFS (2024)
77. Xie, S., Yang, X., Yao, Y., Liu, T., Wang, T., Sun, J.: Label inference attack against split learning under regression setting. arXiv:2301.07284 (2023)
78. Xu, X., Yang, M., Yi, W., Li, Z., Wang, J., Hu, H., Zhuang, Y., Liu, Y.: A stealthy wrongdoer: Feature-oriented reconstruction attack against split learning. In: CVPR (2024)
79. Yang, X., Sun, J., Yao, Y., Xie, J., Wang, C.: Differentially private label protection in split learning. arXiv:2203.02073 (2022)
80. Yu, F., Wang, L., Zeng, B., Zhao, K., Pang, Z., Wu, T.: How to backdoor split learning. Neural Networks (2023)
81. Yu, F., Zeng, B., Zhao, K., Pang, Z., Wang, L.: Chronic poisoning: Backdoor attack against split learning. In: AAAI (2024)
82. Zeng, B., Luo, S., Yu, F., Yang, G., Zhao, K., Wang, L.: Gan-based data reconstruction attacks in split learning. Neural Networks (2025)
83. Zhang, L., Gao, X., Li, Y., Liu, Y.: Functionality and data stealing by pseudo-client attack and target defenses in split learning. IEEE TDSC (2024)
84. Zhao, K., Chuo, X., Yu, F., Zeng, B., Pang, Z., Wang, L.: Splitaum: Auxiliary model-based label inference attack against split learning. IEEE TNSM (2024)
85. Zhu, X., Luo, X., Wu, Y., Jiang, Y., Xiao, X., Ooi, B.C.: Passive inference attacks on split learning via adversarial regularization. arXiv:2310.10483 (2023)

A Details of Backdoor and Poisoning Attacks

Here, we further detail backdoor and poisoning attacks into subcategories.
a. Label Flipping Attacks: Label flipping attacks represent a fundamental poisoning strategy in SL, where adversaries deliberately alter class labels within the training data [20,29,40]. In these attacks, malicious clients tamper with their local datasets before training. Common strategies include targeted label flipping (changing specific source classes to chosen targets), untargeted flipping (random mislabeling), and distance-based optimization (selecting target classes that maximize classification error based on feature similarity). The effectiveness of

label flipping attacks increases with the poisoning rate. For instance, on CIFAR-10, 10% malicious clients cut class recall by 12.6%, and 50% malicious clients by up to 47.3% [20]. Similarly, Ismail and Shukla [29] found that untargeted and distance-based attacks degraded accuracy more than targeted ones. This greater accuracy drop indicates a broader impact because accuracy reflects performance across all classes: targeted attacks disrupt only a specific source–target pair, while untargeted and distance-based strategies corrupt labels across many classes, distorting multiple decision boundaries. When misclassification extends beyond a single class pair, the overall accuracy declines more sharply, confirming that the attack affects the model more widely.

b. Embedding Poisoning Attacks: Embedding poisoning attacks target the smashed data, manipulating the embedding space rather than raw data or labels [4, 76]. The general approach involves introducing perturbations to the feature representations ($\tilde{z}_i$) before they are transmitted to the server. These perturbations are designed to be subtle enough to avoid detection while causing the model to learn unintended patterns or vulnerabilities. These attacks are effective because they directly compromise the information exchange that is fundamental to the collaborative learning process. Based on the [4], embedding poisoning yields an ASR of 1.0 across datasets such as CIFAR-10, and CIFAR-100, while maintaining high CDA, typically in the 0.83–0.90 range, even when poisoning less than 1% of the embeddings.

c. Client-Side Backdoor Attacks: Client-side backdoor attacks in SL involve malicious clients who inject hidden triggers into their local training data or model components. These attacks follow a pattern where the adversary modifies a subset of their training data to include specific trigger patterns associated with targeted misclassifications [63, 80, 81]. The local model is trained to react to triggers while preserving performance on clean data—using feature manipulation, label changes, or auxiliary models to separate clean from backdoored samples. The system's distributed nature obscures malicious behavior, letting backdoors persist across rounds without harming primary task utility. Empirical results show that [81] achieves 92.8% backdoor accuracy on MNIST, 97.4% on Fashion-MNIST, and 91.8% on CIFAR-10, while main-task accuracy remains above 94%, 87%, and 88% respectively.

d. Server-Side Backdoor Attacks: In server-side backdoor attacks, a malicious server compromises the integrity of the SL model. As investigated in [67,80, 81], these attacks leverage the server's privileged position in the training process to implant backdoor functionalities without direct access to client data. This involves manipulating the shared model components or gradients to create hidden vulnerabilities that can be exploited later. This can be achieved through surrogate model training, feature space manipulation, or strategic modification of model updates. Server-side backdoors show high effectiveness [80]: under Split 1, ASR reaches 95.01% on MNIST, 88.22% on Fashion-MNIST, and 96.72% on CIFAR-10, with CDA remaining close to baseline (97.63%, 84.67%, 94.67%).