Information Sciences 719 (2025) 122419

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Information leakage detection through approximate Bayes-optimal prediction

Pritha Gupta [a,b,*], Marcel Wever [c], Eyke Hüllermeier [d,e,f]

[a] Software Innovation Campus Paderborn (SICP), Paderborn University, Paderborn, Germany
[b] Trustworthy Human Language Technologies (TrustHLT), Research Center for Trustworthy Data Science and Security (RCTrust), Ruhr University Bochum, Bochum, Germany
[c] L3S Research Center, Leibniz University Hannover, Hannover, Germany
[d] Institute of Informatics, University of Munich, Munich, Germany
[e] Munich Center for Machine Learning (MCML), Munich, Germany
[f] German Centre for Artificial Intelligence (DFKI/DSA), Kaiserslautern, Germany

ARTICLE INFO

Keywords:
Information leakage detection
Mutual information
Bayes-optimal predictor
AutoML
Statistical tests
Privacy

ABSTRACT

In today's data-driven world, the proliferation of publicly available information raises security concerns due to the information leakage (IL) problem. IL involves unintentionally exposing sensitive information to unauthorized parties via observable system information. Conventional statistical approaches rely on estimating mutual information (MI) between observable and secret information for detecting ILs, face challenges of the curse of dimensionality, convergence, computational complexity, and MI misestimation. Though effective, emerging supervised machine learning based approaches to detect ILs are limited to the binary system, sensitive information, and lacks a comprehensive framework. To address these limitations, we establish a theoretical framework using statistical learning theory and information theory to quantify and detect IL accurately. Using automated machine learning, we demonstrate that MI can be accurately estimated by approximating the typically unknown Bayes predictor's LOG-LOSS and accuracy. Based on this, we show how MI can effectively be estimated to detect ILs. Our method performs superior to state-of-the-art baselines in an empirical study considering synthetic and real-world OpenSSL TLS server datasets.

1. Introduction

The rapid proliferation of publicly available data, coupled with the increasing use of Internet of Things (IoT) technologies in today's data-driven world, has magnified the challenge of IL, posing substantial risks to system security and confidentiality [1]. IL occurs when sensitive or confidential information is inadvertently exposed to unauthorized individuals through observable system information [2]. This can lead to severe consequences, ranging from potential electrical blackouts to the theft of critical information like medical records and military secrets, making the efficient detection and quantification of IL of paramount importance [2,3]. According to information theory, quantifying IL typically involves estimating MI between observable and secret information [4]. Despite being a pivotal measure, MI is difficult to compute for high-dimensional data, facing challenges such as the *curse of dimensionality*, convergence, and computational complexity [5]. Traditional statistical estimation methods often struggle with all of

* Corresponding author.
E-mail addresses: prithag@mail.upb.de (P. Gupta), marcel.wever@ai.uni-hannover.de (M. Wever), eyke@lmu.de (E. Hüllermeier).

https://doi.org/10.1016/j.ins.2025.122419
Received 29 July 2024; Received in revised form 6 June 2025; Accepted 8 June 2025

Available online 13 June 2025

these challenges, while more recent robust non-parametric approaches with improved convergence rates still find high-dimensional scenarios challenging [6].

In recent years, machine learning (ML) techniques have gained popularity in information leakage detection (ILD), particularly for performing side-channel attacks (SCAs) on cryptographic systems [7]. These systems release the *observable information* via many modes called the side-channels, such as network messages, CPU caches, power consumption, or electromagnetic radiation, which are exploited by SCAs to reveal secret inputs (secret keys, plaintexts), potentially rendering cryptographic protections ineffective [8,2]. Therefore, detecting the existence of a side-channel is equivalent to uncovering IL [2]. In this field, the most relevant literature uses ML to perform SCAs rather than preventing side-channels through early detection of ILs [2]. Current ML-based methods in this realm detect side-channels to prevent SCAs and protect the system on both algorithmic and hardware levels [8]. These approaches leverage observable information to classify systems as vulnerable (with IL) or non-vulnerable (without IL) [9]. They extract observable information from secure systems, categorizing them as non-vulnerable (labeled 0), then introduce known ILs to categorize them as vulnerable (labeled 1), creating a classification dataset for the learning model. However, this approach is limited to domain-specific scenarios and cannot be easily transferred to detect other unknown leakages [9].

Recent promising ML-based methods proposed for estimating MI within classification datasets grapple with challenges related to convergence and computational complexity [10], and others may underestimate MI or miss specific subclasses of IL [11]. Recent advancements have demonstrated the effectiveness of ML-based techniques in directly detecting IL by analyzing the accuracy of the supervised learning models on extracted system data [8]. Yet, these methods exhibit limitations in handling imbalanced and noisy real-world datasets, commonly encountered in practical scenarios, and tend to miss ILs by producing false negatives [12,13].

To address these limitations, in our prior work, we proposed utilizing binary classifiers integrated with Fisher's exact test (FET) and paired t-test (PTT) statistical tests to account for imbalance [14]. Despite its merits, this approach is limited to binary classification tasks and needs a comprehensive theoretical framework [14].

Our contributions We address these shortcomings with the following contributions:

- A novel ILD framework with a generalized LAS metric to assess ILs accurately.
- MI estimators approximating Bayes' performance via automated machine learning.
- Bayes' performance in terms of Log-Loss is used to address the data imbalance.
- ILD by thresholding on MI with Holm-Bonferroni correction for robust detection.
- Empirical study of ILD approaches vs. baselines to counter Bleichenbacher SCA.

2. Information leakage detection problem

In this section, we formalize the ILD task of categorizing a system as vulnerable or non-vulnerable using the proposed generalized leakage assessment score (LAS) measure to quantify IL, subsequently used to detect IL in the system. LAS is evaluated by comparing the (approximate) performance of the Bayes predictor and marginal Bayes predictor; when using log-loss, it reduces to MI, a standard measure for IL quantification. We also briefly introduce the concepts of Bayes predictor and MI, with details of statistical learning theory using the notations defined in Table A.4.

2.1. Formal setting

ILD aims to identify unintended disclosure of *secret information* through *observable information* of the system. The ILD algorithm analyzes the system dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$, $N \in \mathbb{N}$, where $\mathcal{X} = \mathbb{R}^d$ represents observable information and $\mathcal{Y} = [C]$ represents secret information as categorical classes. The goal is to labels $\mathcal{D}$ with 1 indicating IL and 0 its absence, denoted by mapping L as

$$L : \bigcup_{N \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^N \to \{0, 1\},$$

which takes a dataset $\mathcal{D}$ of any size as input and outputs the decision on the presence of IL in the system. The ILD approach produces the mapping $\hat{L}$ and predicts ILs in the given system. Let $\mathcal{L} = \{(\mathcal{D}_i, z_i)\}_{i=1}^{N_L}$ be an **IL-Dataset**, such that $N_L \in \mathbb{N}, z_i \in \{0, 1\}$ and $\boldsymbol{z} = (z_1, \ldots, z_{N_L})$ be the ground truth vector generated by L. The predicted ILs produced by $\hat{L}$ are denoted as the vector $\hat{\boldsymbol{z}} = (\hat{z}_1, \ldots, \hat{z}_{N_L})$, such that $\hat{z}_i = \hat{L}(\mathcal{D}_i)$. The performance of an ILD approach ($\hat{L}$) is measured using classification metrics ($\mathrm{m}_{(\cdot)}(\boldsymbol{z}, \hat{\boldsymbol{z}})$) (cf. Appendix A.2.1).

2.2. Fundamentals

We briefly introduce the concepts of Bayes predictor and MI, including the details of the classification problem.

2.2.1. Mutual information

Mutual information (MI) measures the extent to which knowledge of one random variable informs about another, quantifying their dependence degree [15]. Consider a pair of random variables X and Y with joint distribution $p_{(X,Y)}(\cdot)$ on $\mathcal{X} \times \mathcal{Y}$. We assume that X is a continuous d-dimensional real-valued random variable ($\mathcal{X} = \mathbb{R}^d$), and Y is a discrete random variable with C possible values—as in the IL scenario relevant to us [4]. Let the measure P induces a marginal probability density function (PDF) on $\mathcal{X}$ and $\mathcal{Y}$ denoted by $p_X(\cdot)$ and $p_Y(\cdot)$ of the joint distribution $p_{(X,Y)}(\cdot)$, which induces the conditional distributions $p_{Y|X}(\cdot)$ and $p_{X|Y}(\cdot)$.

The entropy of a discrete random variable Y is defined as

$$H(Y) = -\sum_{y\in\mathcal{Y}} p_Y(y)\log(p_Y(y))\,, \tag{1}$$

where $0\log(0) = 0$ by definition. It reaches the maximum value of $\log(C)$ when outcomes are equally likely ($p_Y(y) = 1/C, \forall y \in [C]$), indicating complete uncertainty of the outcome. The minimum value of 0 occurs in the case of a Dirac measure when only one outcome is certain, i.e., $p_Y(y) = 0, \forall y \in [C] \setminus c$, $p_Y(c) = 1$.

The conditional entropy of Y given X is defined as

$$H(Y \mid X) = -\int_{\boldsymbol{x}\in\mathcal{X}} p_X(\boldsymbol{x}) \sum_{y\in\mathcal{Y}} p_{Y|X}(y \mid \boldsymbol{x})\log(p_{Y|X}(y \mid \boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}\,.$$

Conditional entropy measures the residual uncertainty in one random variable Y given knowledge of the other X — more specifically, it measures the *expected* residual uncertainty, with the expectation taken with respect to the marginal distribution of Y. It reaches its maximum value of $H(Y)$, when X does not inform about Y, i.e., $H(Y \mid \boldsymbol{x}) = p_Y(\cdot), \forall \boldsymbol{x} \in \mathcal{X}$, and the minimum entropy of 0 occurs when X completely determines Y and again all conditionals $p_{Y|X}(\cdot \mid \boldsymbol{x})$ are Dirac distributions.

MI measures the reduction of uncertainty about variable Y by observing variable X:

$$I(X;Y) = H(Y) - H(Y \mid X)\,. \tag{2}$$

Plugging in the expressions for (conditional) entropy and rearranging terms shows that MI equals the Kullback-Leibler (KL) divergence of the joint distribution $p_{(X,Y)}(\cdot)$ from the product of the marginals (i.e., the joint distribution under the assumption of independence):

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y \mid X) \\ &= \int_{\boldsymbol{x}\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{(X,Y)}(\boldsymbol{x}, y)\log\left(\frac{p_{(X,Y)}(\boldsymbol{x}, y)}{p_X(\boldsymbol{x})\cdot p_Y(y)}\right) d\boldsymbol{x}\,. \end{aligned} \tag{3}$$

MI is a symmetric measure ranging from 0 to $\min(\{H(X), H(Y)\})$, where 0 indicates independence and the maximum value signifies full dependence [15]. In this paper, MI is measured in ***bits*** using base-2 logarithms ($\log(\cdot)$).

2.2.2. Classification problem

In the realm of classification, the learning algorithm (learner) is provided with a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \subset \mathcal{X}\times\mathcal{Y}$ of size $N \in \mathbb{N}$, where $\mathcal{X} = \mathbb{R}^d$ is the input (instance) space and $\mathcal{Y} = \{1, 2, \ldots, C\} = [C]$, $C \in \mathbb{N}$ the output (categorical classes) space [16], and the $(\boldsymbol{x}_i, y_i)$ are assumed to be independent and identically distributed (i.i.d.) according to $p_{(X,Y)}(\cdot)$. The primary goal of the learner in standard classification is to induce a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, $h \in \mathcal{H}$, with low generalization error (risk)

$$R(h) = \mathbb{E}[\ell(y, h(\boldsymbol{x}))] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h(\boldsymbol{x}))\, d\, p_{(X,Y)}(\boldsymbol{x}, y)\,, \tag{4}$$

where $\mathcal{H}$ is the underlying hypothesis space, $\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$ is a loss function, and $p_{(X,Y)}(\cdot)$ is the joint probability measure modeling the underlying data-generating process. The risk minimizer defined as $h^* = \arg\min_{h\in\mathcal{H}} R(h)$, achieves the minimum expected loss $\mathbb{E}[\ell(\cdot)]$ in terms of the loss function ℓ across the entire joint distribution $p_{(X,Y)}(\cdot)$. The 0-1 loss defined as $\ell_{01}(y, \hat{y}) := [\![\hat{y} \neq y]\!]$, is commonly used in standard classification. In practice, the $p_{(X,Y)}(\boldsymbol{x}, y)$ is not directly observed by the learner, so minimizing the risk (4) is not feasible. Instead, learning in a standard classification setting is commonly accomplished by minimizing (a regularized version of) *empirical risk* for h:

$$R_{\text{emp}}(h) = \frac{1}{N}\sum_{i=1}^N \ell(y_i, h(\boldsymbol{x}_i))\,. \tag{5}$$

In the subsequent discussions, we denote by $g = \arg\min_{h\in\mathcal{H}} R_{\text{emp}}(h)$ the (learned) hypothesis that minimizes (5), i.e., the empirical risk minimizer, which in principle is the best possible approximation (empirical estimation) h^* and is an approximation thereof [16].

Probabilistic classification Distinct from standard classifiers, probabilistic classifiers focus on estimating the (conditional) class probabilities $p_{Y|X}(y \mid \boldsymbol{x})$ for each class y in $\mathcal{Y}$, for a given input instance $\boldsymbol{x} \in \mathcal{X}$. We denote predictions of that kind by $\hat{p}_{Y|X}(y \mid \boldsymbol{x})$. As before, training data comes in the form $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \subset \mathcal{X}\times\mathcal{Y}$, where the $(\boldsymbol{x}_i, y_i)$ are i.i.d. according to $p_{(X,Y)}(\cdot)$, and the goal to induce a hypothesis $h_p : \mathcal{X} \to \mathbb{P}(\mathcal{Y})$ with low generalization error (risk):

$$R_p(h_p) = \mathbb{E}[\ell_p(y, h_p(\boldsymbol{x}))] = \int_{\mathcal{X}\times\mathcal{Y}} \ell_p(y, h_p(\boldsymbol{x}))\, d\, p_{(X,Y)}(\boldsymbol{x}, y)\,.$$

Now, however, instead of comparing a predicted class $\hat{y}$ with a true class y, the loss ℓ_p compares a predicted probability distribution with y — thus, the loss is a mapping $\ell_p : \mathcal{Y}\times\mathbb{P}(\mathcal{Y}) \to \overline{\mathbb{R}}$, where $\mathbb{P}(\mathcal{Y})$ denotes the set of probability mass functions (PMFs) on $\mathcal{Y}$. The

most commonly used loss function $\ell_p(\cdot)$ is the categorical cross-entropy (CCE), defined as $\ell_{\text{CCE}}(y, \hat{\boldsymbol{p}}) := -\ln(\hat{p}_y)$ [17, chap. 4]. The CCE loss is a proper scoring rule widely recognized for its information-theoretic interpretations and practical effectiveness [18].

Since $\mathcal{Y}$ is finite and consists of C classes, $\mathbb{P}(\mathcal{Y})$ can be represented by the $(C-1)$-simplex, i.e., predictions can be represented as probability vectors $h_p(\boldsymbol{x}) = \hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_C)$, where $\hat{p}_c = \hat{\boldsymbol{p}}[c] = \hat{p}_{Y|X}(c \mid \boldsymbol{x})$ is the probability assigned to class c. Typically, learning predictors of that kind involves minimizing the *empirical risk*

$$R_{\text{emp}|p}(h_p) = \frac{1}{N} \sum_{i=1}^{N} \ell_p(y_i, h_p(\boldsymbol{x}_i)) \,. \tag{6}$$

In the subsequent discussions, we denote the empirical risk minimizer by $g_p = \arg\min_{h_p \in \mathcal{H}_p} R_{\text{emp}|p}(h_p)$ [16,17, chap. 4]. In the case where Y is independent of X, the marginal Bayes predictor or marginal classifier (denoted by g_p^{mc}) is again the best constant probability predictor, i.e., the one with the lowest risk (6) among all constant predictors. Obviously, a probabilistic prediction can be used deterministically by selecting the class with the highest predicted probability: $\hat{y} = \arg\max_{y \in \mathcal{Y}} \hat{p}_{Y|X}(y \mid \boldsymbol{x})$ to induce a deterministic classifier g. This approach minimizes the standard 0-1 loss on average, emphasizing the importance of accurately identifying the most probable class for optimal performance [17, chap. 4].

2.2.3. Bayes-optimal predictor

In statistical learning theory, the *Bayes predictor* is the optimal classification function $g^{\text{b}} : \mathcal{X} \to \mathcal{Y}$, which minimizes expected risk (4) for a given loss function $\ell(\cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$:

$$g^{\text{b}}(\boldsymbol{x}) = \arg\min_{\hat{y} \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} \ell(y, \hat{y}) \cdot p_{Y|X}(y \mid \boldsymbol{x}) = \arg\min_{\hat{y} \in \mathcal{Y}} \mathbb{E}_y[\ell(y, \hat{y}) \mid \boldsymbol{x}] \,,$$

where $\mathbb{E}_y[\ell]$ is the expected loss of prediction $\hat{y}$ for $y \in \mathcal{Y}$, and $p_{Y|X}(y \mid \boldsymbol{x})$ is the (conditional) class y probability for input $\boldsymbol{x}$ [19]. For 0-1 loss, it simplifies to:

$$g^{\text{bc}}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} p_{Y|X}(y \mid \boldsymbol{x}) \,. \tag{7}$$

It produces the minimum expected loss, known as *Bayes error rate*, denoted by $\text{m}_{\text{ERR}}(g^{\text{bc}})$. When $\mathcal{X}$ and $\mathcal{Y}$ are independent of each other, i.e., $p_{Y|X}(y \mid \boldsymbol{x}) = p_Y(y), \forall \boldsymbol{x} \in \mathcal{X}$, it reduces to the marginal Bayes predictor:

$$g^{\text{bc}}(\boldsymbol{x}) \equiv g^{\text{mc}}(\boldsymbol{x}) \equiv \arg\max_{y \in \mathcal{Y}} p_Y(y) \,. \tag{8}$$

It assigns to each input $\boldsymbol{x}$ a class label from the set of labels with the highest marginal probability, as input features are completely uninformative. Formally, as the maximum in (8) is not necessarily unique, the marginal Bayes predictor may pick any label with the highest probability — in that case, we nevertheless assume that it picks the same label for every $\boldsymbol{x}$, so that it is a constant function.

2.3. Quantifying and detecting information leakage

IL occurs when observable information ($\boldsymbol{x} \in \mathcal{X}$, represented by X) is correlated with secret information ($y \in \mathcal{Y}$, represented by Y), allowing inference of y from $\boldsymbol{x}$ [1,4]. To quantify IL, we introduce LAS $\delta(\cdot)$, evaluating the difference in average penalties of marginal Bayes predictor and Bayes predictor using loss functions ($\ell_{(\cdot)}$) or metrics ($\text{m}_{(\cdot)}$):

$$\delta(\text{m}_{(\cdot)}) = |\text{m}_{(\cdot)}(g^{\text{mc}}) - \text{m}_{(\cdot)}(g^{\text{bc}})|, \;\; \delta(\ell_{(\cdot)}) = \ell_{(\cdot)}(g^{\text{mc}}) - \ell_{(\cdot)}(g^{\text{bc}}) \,,$$

where $|\cdot|$ is used to avoid negative values for accuracy measures. When using loss function as log-loss, LAS is equal to **MI**, i.e., $\delta(\ell_{ll}) = I(X;Y) = \mathbb{E}[\ell_{ll}(g^{\text{mc}})] - \mathbb{E}[\ell_{ll}(g^{\text{bc}})]$ and Bayes error rate ($\text{m}_{\text{ERR}}(g^{\text{bc}})$) bounds the MI, which serves as the foundation for our LOG-LOSS and MID-POINT estimation approaches, respectively, as discussed in Section 3.1.

In practice, since $p_{(X,Y)}(.)$ is seldom observed, we approximate LAS using empirical risk minimizers (g_p or g) as proxies for Bayes predictor and g_p^{mc} for marginal Bayes predictor:

$$\delta(\text{m}_{(\cdot)}) \approxeq |\text{m}_{(\cdot)}(g_p^{\text{mc}}) - \text{m}_{(\cdot)}(g)|, \;\; \delta(\ell_{(\cdot)}) \approxeq \ell_{(\cdot)}(g_p^{\text{mc}}) - \ell_{(\cdot)}(g_p) \,.$$

For losses evaluated using (conditional) class probabilities, g_p minimizing (6) is used as a proxy for the Bayes predictor. However, for decision-based classification losses, g minimizing (5) or the decision rule of g_p is used, with g being a better proxy for error rate or accuracy measure.

IL detection IL occurs if LAS is significantly greater than 0, i.e., $\delta(\text{m}_{(\cdot)}) \gg 0$ or $\delta(\ell_{(\cdot)}) \gg 0$. Thus, ILD involves analyzing the learnability of empirical risk minimizers (g_p or g) on the dataset $\mathcal{D}$ used for the system. Our prior work [14] introduced classification-based ILD approaches using an average error rate (0-1 loss) to quantify and detect IL by analyzing $\delta(\text{m}_{\text{ERR}})$ and $\delta(\text{m}_{\text{CM}})$ using PTT and FET, respectively. We also propose MI-based ILD approaches using the one-sample t-test (OTT) on MI estimates, with CAL LOG-LOSS effectively detecting IL in OpenSSL TLS servers (cf. Section 4.3).

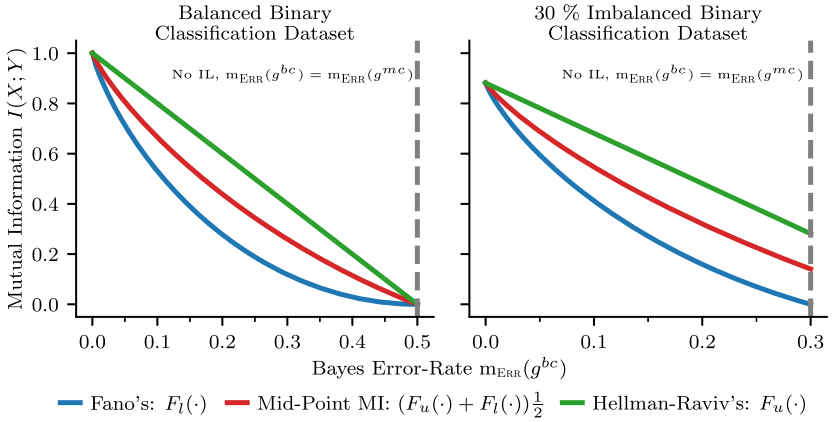


Fig. 1. Bounding MI (y-axis) with respect to Bayes error rate (x-axis) for a balanced and imbalanced binary classification dataset with 30 % of minority classes (*class imbalance* level $r = 0.3$). The upper bound $F_u(\cdot)$ is derived using $H_l(\cdot)$ by Hellman and Raviv [23] and the lower bound $F_l(\cdot)$ is defined using the upper bound $H_u(\cdot)$ derived by Fano [24]. The gray lines represent the case of no IL, where the Bayes-optimal predictor converges to the marginal Bayes predictor.

3. Mutual information estimation

This section introduces two methods for estimating MI in classification datasets. We compare their performance on synthetic datasets against state-of-the-art approaches including mutual information neural estimation (MINE) [20], Gaussian mixture model (GMM) [21], and PC-Softmax [11]. While MINE and GMM do not explicitly handle imbalance—MINE which was initially proposed to estimate MI for two real-valued vectors, the classes are binary encoded to estimate MI and have already been proposed to assess IL, and GMM fits per-class Gaussians, which are widely used state-of-the-art approaches [10]. In contrast, PC-Softmax addresses class imbalance by incorporating label frequency to modify CCE loss function to propose probability-corrected softmax (PC-softmax) to estimate MI considering the $p_Y(Y)$, making it a fair and relevant baseline for comparison with Log-Loss [11]. We describe the experimental setup, including the configuration of datasets generated by simulated systems using the multivariate normal (MVN) distribution and evaluation metrics, and summarize the results in Section 3.3. Our results demonstrate the superior performance of our proposed methods for estimating MI.

3.1. Our approaches

We introduce two methods for estimating MI in classification datasets called Log-Loss, which explicitly targets imbalanced datasets, and Mid-Point, which estimates MI using only model accuracy, rather than relying solely on standard statistical methods.

3.1.1. Mid-point estimation

The Mid-Point approach estimates MI by leveraging the relationship between Bayes error rate $\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}})$ and the conditional entropy $H(Y \mid X)$ for a classification task [22]. The MI is estimated using the empirical risk minimizer g of (5) to approximate the Bayes error rate.

The conditional entropy $H(Y \mid X)$ is bounded as

$$
\begin{aligned}
&H_l(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C) \le H(Y \mid X) \le H_u(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C)\\
&H_l(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C) = \log(c) + c(c+1)\left(\log\left(\frac{c+1}{c}\right)\right)\left(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}) - \frac{c-1}{c}\right)\\
&H_u(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C) = H_2(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}})) + \mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}) \cdot \log(C-1),
\end{aligned}
$$

where $H_l(\cdot)$ derived by Hellman and Raviv [23] is valid for $\frac{1}{c+1} \le 1 - \mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}) \le \frac{1}{c}, c = 1, \ldots, C-1$ and $H_u(\cdot)$ derived by Fano [24] uses the binary cross-entropy function, $H_2(a) = -a\log(a) - (1-a)\log(1-a)$.

Plugging in $H_l(\cdot)$ and $H_u(\cdot)$ in (2), the bounds on MI the lower bound $F_l(\cdot)$ and the upper bound $F_u(\cdot)$ with respect to Bayes error rate and the number of classes C is derived as

$$
\begin{aligned}
&H(Y) - H_u(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C) \le I(X;Y) \le H(Y) - H_l(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C)\\
&F_l(\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), C) \le I(X;Y) \le F_u(\mathrm{m}_{\mathrm{ERR}}(,)g^{\mathrm{bc}}, C).
\end{aligned}
$$

MI is estimated as

$$
\widehat{I}(X;Y) \approxeq \frac{F_u(\mathrm{m}_{\mathrm{ERR}}(g), C) + F_l(\mathrm{m}_{\mathrm{ERR}}(g), C)}{2}, \tag{9}
$$

where g is the *empirical risk minimizer* of (5) and serves as a proxy of the Bayes predictor, as $\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}) \approxeq \mathrm{m}_{\mathrm{ERR}}(g)$ (Fig. 1).

Overestimation in imbalanced data To illustrate the issue of MI overestimation by the MID-POINT approach—leading to false positives in IL detection—we visualize MI estimated using MID-POINT in a *non-vulnerable* system that generates imbalanced binary classification data with a class *imbalance level* of 30 % ($r = 0.3$, defined in Appendix A.1.1) in Fig. 1. The *non-vulnerable* system corresponds to the condition where the Bayes-optimal predictor converges to the marginal Bayes predictor, as discussed in Section 2.3. While Zhao et al. [25] relates metrics like balanced error rate (BER) to the conditional entropy $H(Y \mid X)$ in binary classification, such relationships do not extend naturally to the multi-class case.

3.1.2. LOG-LOSS *estimation*

We propose the LOG-LOSS approach for MI estimation to address the overestimation and false positives of the MID-POINT approach and the false negatives of baseline ILD approaches. This approach uses the empirical risk minimizer g_p minimizing (6) to approximate the log-loss of the Bayes predictor.

The MI between X and Y in (2) is defined as

$$\begin{aligned} I(X;Y) &= -H(Y \mid X) + H(Y) \\ &= \int_{\boldsymbol{x}\in\mathcal{X}} p_X(\boldsymbol{x}) \sum_{y\in\mathcal{Y}} p_{Y|X}(y \mid \boldsymbol{x}) \log\left(p_{Y|X}(y \mid \boldsymbol{x})\right) d\boldsymbol{x} - \sum_{y\in\mathcal{Y}} p_Y(y) \log\left(p_Y(y)\right) \\ &= -\mathbb{E}[\ell_{ll}(g^{\mathrm{bc}})] + \mathbb{E}[\ell_{ll}(g^{\mathrm{mc}})] \approxeq -\mathbb{E}[\ell_{ll}(g_p)] + \mathbb{E}[\ell_{ll}(g_p^{\mathrm{mc}})], \end{aligned}$$

where $\mathbb{E}[\ell_{ll}(\cdot)]$ represents the expected log-loss. The conditional entropy $H(Y \mid X)$ equals the expected log-loss of the Bayes predictor g^{bc}, while the entropy of Y corresponds to the expected log-loss of the marginal Bayes predictor g^{mc}, reaffirming MI as a special case of LAS (cf. Section 2.3).

Log-loss For probabilistic classifiers g_p, CCE is used to obtain estimated conditional class probabilities, forming a vector $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_C)$ for each input $\boldsymbol{x}$. The log-loss for g_p is defined in Bishop [17, chap. 4] as

$$\ell_{ll}(y, \hat{\boldsymbol{p}}) = -\sum_{c=1}^{C} \hat{p}_c \log(\hat{p}_c) = -\sum_{c=1}^{C} g_p(\boldsymbol{x})[c] \log(g_p(\boldsymbol{x})[c]).$$

It reaches its minimum value of 0 when one class dominates, i.e., $\hat{p}_c \approxeq 1$ and $\hat{p}_j \approxeq 0$ for all $j \in [C] \setminus \{c\}$, and its maximum $\log(C)$ when class probabilities are uniform, i.e., $\hat{p}_c = 1/C$ for all $c \in [C]$, implying that high classification uncertainty increases the log-loss, making it suitable for estimating the conditional entropy $H(Y \mid X)$.

Marginal Bayes predictor It is estimated from the class distribution in the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ using (6) and is denoted by g_p^{mc}. The class probabilities for each input $\boldsymbol{x}$ are $g_p^{\mathrm{mc}}(\boldsymbol{x}) = \hat{\boldsymbol{p}}_{mc} = (\hat{p}_1, \ldots, \hat{p}_C)$, where $\hat{p}_c = \frac{|\{(\boldsymbol{x}_i, y_i) \in \mathcal{D} \mid y_i = c\}|}{|\mathcal{D}|}$ is the fraction of instances for class c. The log-loss of the marginal Bayes predictor suitably estimates the entropy of Y, i.e., $H(Y) \approxeq \ell_{ll}(y, \hat{\boldsymbol{p}}_{mc})$, ranging from 0 to $\log(C)$, with the maximum acquired for a balanced dataset.

MI estimation We approximate the expected log-loss of the Bayes predictor and marginal Bayes predictor by evaluating the log-loss of g_p and g_p^{mc} for the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ as

$$\mathbb{E}[\ell_{ll}(g_p)] = \frac{1}{N}\sum_{i=1}^{N} \ell_{ll}(y_i, g_p(\boldsymbol{x}_i)) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} g_p(\boldsymbol{x}_i)[c] \log(g_p(\boldsymbol{x}_i)[c])$$

$$\mathbb{E}[\ell_{ll}(g_p^{\mathrm{mc}})] = \frac{1}{N}\sum_{i=1}^{N} \ell_{ll}(y_i, g_p^{\mathrm{mc}}(\boldsymbol{x}_i)) = -\sum_{c=1}^{C} \hat{\boldsymbol{p}}_{mc}[c] \log(\hat{\boldsymbol{p}}_{mc}[c]).$$

The MI is then estimated as

$$\hat{I}(X;Y) \approxeq \sum_{c=1}^{C} \left(\frac{1}{N}\sum_{i=1}^{N} g_p(\boldsymbol{x}_i)[c] \log(g_p(\boldsymbol{x}_i)[c]) - \hat{\boldsymbol{p}}_{mc}[c] \log(\hat{\boldsymbol{p}}_{mc}[c]) \right). \tag{10}$$

Classifier calibration Despite the sound theoretical grounding of the proper scoring rule loss function, CCE often yields poorly calibrated probabilities in practice, with neural networks tending to be overconfident, especially on imbalanced datasets [26]. Post-hoc *calibration* techniques address this by learning mappings from predicted to improved probabilities using validation data [27]. To enhance LOG-LOSS estimation accuracy, we employ calibration techniques like Isotonic Regression (IR CAL LOG-LOSS), Platt's Scaling (PS CAL LOG-LOSS), Beta Calibration (BETA CAL LOG-LOSS), Temperature Scaling (TS CAL LOG-LOSS), and Histogram Binning (HB CAL LOG-LOSS), referred collectively as CAL LOG-LOSS [27]. These calibration techniques are pivotal for enhancing LOG-LOSS, and enabling accurate MI estimation—thereby avoiding the false positives often observed with the MID-POINT approach.

Table 1
Overview of the synthetic datasets for MI estimation experiments.

MVN Perturbation & MVN Proximity Synthetic Datasets					
Dataset Type	Generation Method (g_r)	Input Dimensions (d)	Classes (C)	Noise Level (ϵ)	Class Imbalance (r)
Balanced	NA	{2, 4, …, 20}	{2, 4, …, 10}	{0.0, 0.1, …, 1.0}	NA
Imbalanced Binary-class	Minority	5	2	{0.0, 0.1, …, 1.0}	{0.05, 0.1, …, 0.5}
Imbalanced Multi-class	Minority, Majority	5	5	{0.0, 0.1, …, 1.0}	{0.02, 0.04, …, 0.2}

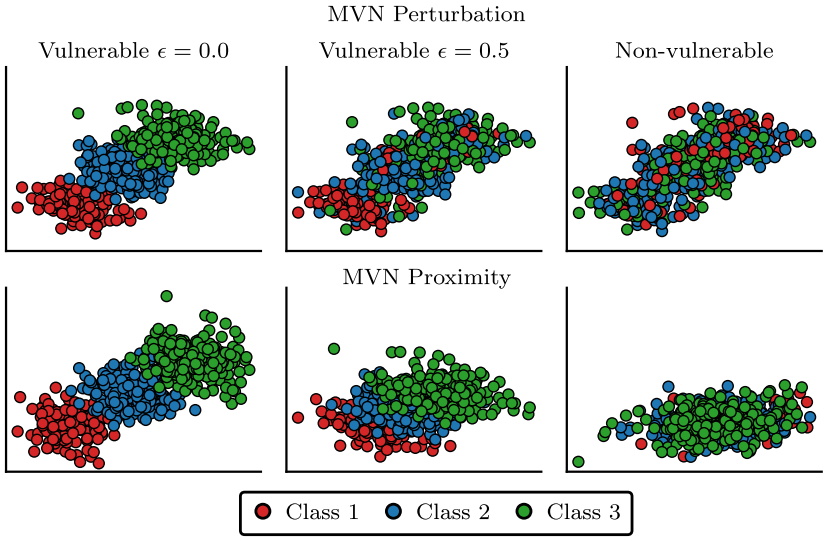


Fig. 2. Visualization of three-class synthetic datasets in 2-D space, generated using MVN perturbation and proximity-based techniques to simulate ***vulnerable*** systems ($\epsilon = 0, 0.5$) and a ***non-vulnerable*** system ($\epsilon = 1.0$).

3.2. Empirical evaluation

This section outlines the evaluation process for our MI estimation methods compared to baselines, as illustrated in Fig. 4. Our goal is to assess the generalization capabilities of these approaches under various conditions, including the number of classes (C), input dimensions (d), class imbalance (r), and noise level (ϵ), using datasets generated by systems with different configurations (g_r, C, d, r, ϵ) outlined in Table 1. Overall and generalization performance results are discussed in Section 3.3.

3.2.1. Synthetic datasets

To evaluate our MI estimation methods, we generated synthetic datasets by simulating real-world systems using the MVN distribution, allowing for straightforward calculation of ground truth MI, discussed in Appendix A.1.1. MVN perturbation and proximity techniques were used to introduce noise and simulate systems with varying levels of vulnerability, as detailed in Appendix A.1.2. The perturbation technique introduces noise by flipping a percentage of class labels, and the proximity technique reduces the distance between the mean vectors of the MVN for each class, causing the generated Gaussians to overlap. To simulate the realistic scenarios, we define the ***vulnerable systems*** as those with noise levels $\epsilon \in [0.0, 0.90)$, where side-channel signals remain distinguishable, and ***non-vulnerable systems*** as those with $\epsilon \in [0.90, 1.0]$, where high noise renders secret information statistically indistinguishable using the side-channel observable information.

This assumption is supported by the **Marvin SCA** [28], which shows that even noisy side-channels can be exploited using statistical tests and system-level optimizations. Fig. 2 illustrates two-dimensional data points with class *imbalance level* ($r = 0.3$), generated using both MVN perturbation and proximity-based techniques. The datasets include ***vulnerable*** systems at noise levels $\epsilon = 0$ and $\epsilon = 0.5$, and a ***non-vulnerable*** system at $\epsilon = 1.0$. We introduce *class imbalance* (r) using *Minority* and *Majority* sampling strategies to define the output prior $p_Y(\cdot)$, as described in Appendix A.1.1, with example class distributions for each case of imbalanced binary-class and multi-class datasets illustrated in Fig. 3. To evaluate MI estimation accuracy of different approaches under varying class distributions (class imbalance levels r), we generated datasets for balanced, imbalanced binary-class, and multi-class scenarios. Each dataset configuration varies in input dimension (d), number of classes (C), class imbalance level (r), and noise level (ϵ), as summarized in Table 1. All datasets were independently generated across the full noise range ($\epsilon \in [0.0, 1.0]$) to simulate both ***vulnerable*** ($\epsilon \in [0.0, 0.90)$) and ***non-vulnerable*** ($\epsilon \in [0.90, 1.0]$) systems.

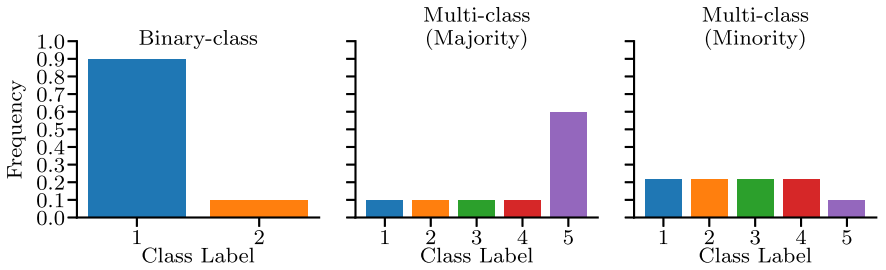


Fig. 3. Classification dataset class distribution with 10 % of *class imbalance* level ($r = 0.1$).

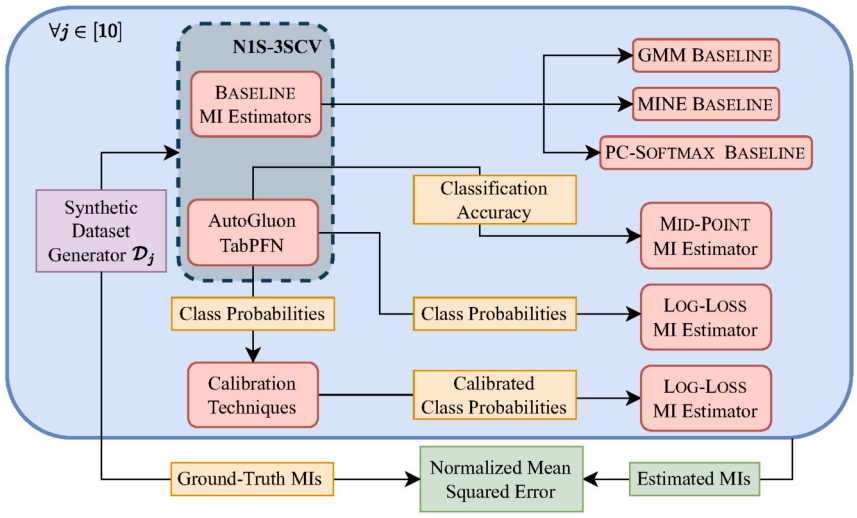


Fig. 4. Experimental setup for evaluating MI estimation methods.

3.2.2. Experimental setup

We evaluate each MI estimation approach on synthetic datasets generated for configurations outlined in Table 1, as shown in Fig. 4. For each configuration $(\mathrm{g}_r, C, d, r, \epsilon)$, we generate 10 datasets $(\mathcal{D}_j)$ using different seeds $j \in [10]$ and evaluate the performance using normalized mean absolute error (NMAE). Using nested cross-validation with hyperparameter optimization (HPO), we split $\mathcal{D}_j$ into 70 % training and 30 % test datasets. HPO involves 100 function evaluations using Monte Carlo cross-validation (MCCV) with 3 splits on the training set, reserving 30 % for validation, denoted as "N1S-3SCV", as shown in Fig. 4. Objective functions for HPO include BER for PC-Softmax, AutoGluon, and TabPFN, Akaike information criterion (AIC) for GMM, and mean squared error (MSE) for MINE with parameter ranges provided in Table A.5. We identify the best-performing pipeline from running AutoGluon for 1800 seconds and model using HPO on GMM, MINE, PC-Softmax, and TabPFN using validation loss or accuracy. The estimated MI of this model on the dataset $\mathcal{D}_j$, is compared with respect to the ground truth MI in (A.8), denoted by $\hat{I}_j$ and I_j, using the NMAE in (11).

3.2.3. Evaluation metric

We assess generalization using normalized mean absolute error (MAE) (NMAE), computed by dividing MAE by the entropy of Y, i.e., $H_Y(\mathrm{g}_r, C, r) = -\sum_{c=1}^{C} p_Y(c) \log\left(p_Y(c)\right)$, since MI ranges from 0 to $\log(C)$. The NMAE metric for 10 ground-truth and estimated MI values is defined as:

$$\mathrm{m}_{\mathrm{NMAE}}(\boldsymbol{I}, \hat{\boldsymbol{I}}) = \frac{1}{10} \sum_{j=1}^{10} \frac{|\boldsymbol{I}[j] - \hat{\boldsymbol{I}}[j]|}{H_Y(\mathrm{g}_r, C, r)} = \sum_{j=1}^{10} \frac{|I_j - \hat{I}_j|}{10 \cdot H_Y(\mathrm{g}_r, C, r)}. \tag{11}$$

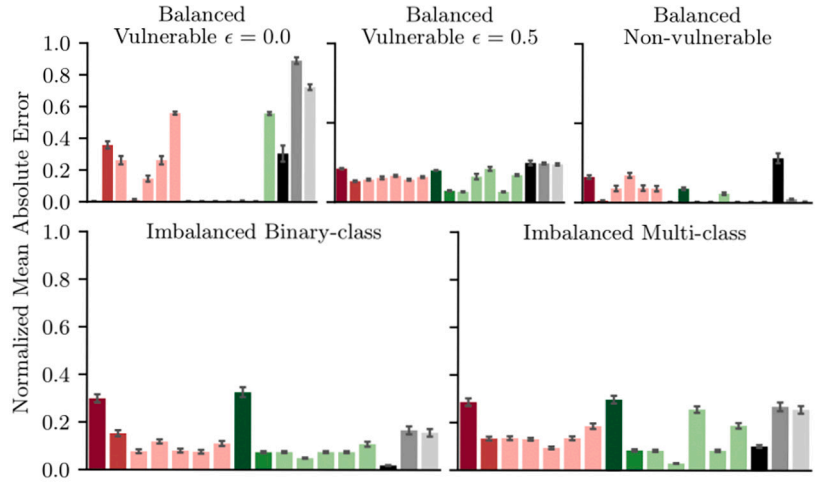
3.3. Results

This section discusses the overall performance of various MI estimation methods on systems simulated using perturbation and proximity techniques. We also analyze the generalization capabilities of selected approaches across various factors, including the number of classes (C), input dimensions (d), class imbalance (r), and noise levels (ϵ). Our analysis covers balanced, binary-class, and imbalanced multi-class datasets, as detailed in Table 1. We examine datasets generated by simulated ***vulnerable systems*** with noise levels of 0 % ($\epsilon = 0.0$, representing a fully compromised system) and 50 % ($\epsilon = 0.5$, representing a compromised system with ineffective countermeasures against SCAs). In contrast, datasets from ***non-vulnerable systems*** are simulated with noise levels $\epsilon \in [0.90, 1.0]$, representing systems robust even against advanced attacks such as the Marvin SCA, which can exploit noisy information to reveal

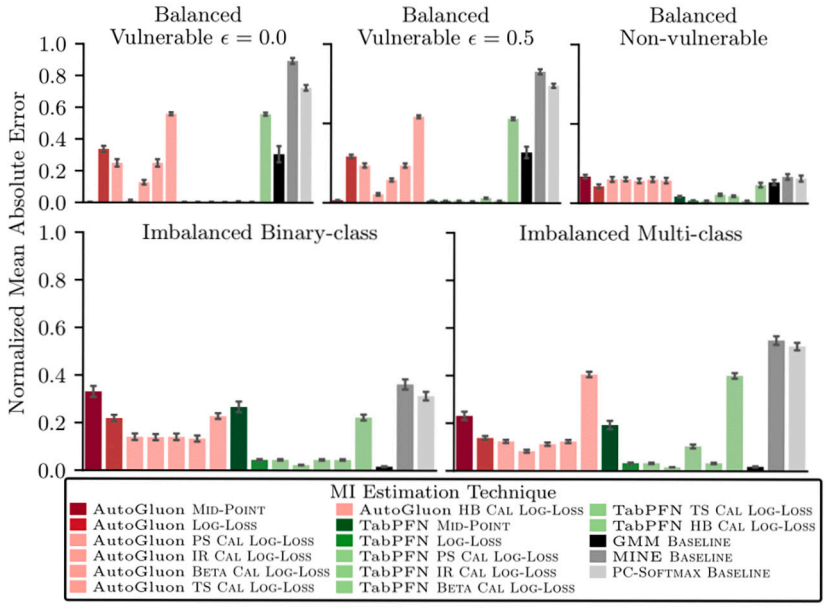
secrets [28]. This setup highlights the importance of accurate MI estimation for effective ILD. Additionally, to assess accurate MI estimation for ILD in imbalanced datasets from realistic systems, we examine the binary and multi-class imbalanced datasets generated by various vulnerable systems ($\epsilon \in [0.00, 0.90)$) and two non-vulnerable systems with $\epsilon \in \{0.90, 1.0\}$.

3.3.1. Overall performance

We present the performance of various MI estimation methods using bar charts showing mean and standard error (SE) of NMAE on systems simulated by MVN perturbation and proximity techniques, as shown in Figs. 5a and 5b, respectively.



(a) Performance on systems simulated using MVN perturbation technique



(b) Performance on systems simulated using MVN proximity technique

Fig. 5. Overall NMAE of MI estimation methods on synthetic datasets

MVN perturbation The results depicted in Fig. 5a indicate that TabPFN IR Cal Log-Loss consistently excels in estimating MI, and methods using AutoGluon significantly underperform compared to those using TabPFN. For balanced datasets, TabPFN IR Cal Log-Loss shows exceptional performance (≈ 0.003) and low SE of 0.001. For more vulnerable systems with noise ($\epsilon = 0.5$), TabPFN IR Cal Log-Loss performs less effectively ($\approx 0.212 \pm 0.012$), while TabPFN PS Cal Log-Loss and Beta Cal Log-Loss perform better ($\approx 0.067 \pm 0.003$). The HB Cal Log-Loss method does not enhance the Log-Loss estimation for both automated machine learning (AutoML) tools. For systems generating *imbalanced binary-class* datasets, the GMM shows exceptional performance ($\approx 0.019 \pm 0.002$), with TabPFN Log-Loss and TabPFN IR Cal Log-Loss also performing well (≈ 0.05). For *imbalanced multi-class* datasets, TabPFN IR

CAL LOG-LOSS leads ($\approx 0.029 \pm 0.001$), with the GMM also performing well ($\approx 0.10 \pm 0.002$), demonstrating strong adaptability for imbalanced datasets.

MVN proximity Overall, MI estimation is more straightforward on datasets generated using proximity than on ones using the perturbation technique, as it introduces more complexity (Appendix A.1). However, performance trends are similar to MVN perturbation datasets: TabPFN IR CAL LOG-LOSS outperforms, while AutoGluon performs worse. For systems generating *balanced* datasets, TabPFN IR CAL LOG-LOSS performs exceptionally well for vulnerable systems (≈ 0.003 for $\epsilon = 0.0$ and ≈ 0.010 for $\epsilon = 0.5$) with low variance. For non-vulnerable systems, TabPFN PS CAL LOG-LOSS and TS CAL LOG-LOSS perform best (≈ 0.016). For systems generating *imbalanced binary-class* datasets, the GMM and TabPFN IR CAL LOG-LOSS are top performers ($\approx 0.017 \pm 0.001$). In *imbalanced multi-class* datasets, TabPFN IR CAL LOG-LOSS maintains superior performance (≈ 0.016), while the GMM also performs well (≈ 0.017).

Summary The TabPFN approaches, particularly IR CAL LOG-LOSS, consistently excel in MI estimation, with AutoGluon approaches significantly underperforming compared to those using TabPFN. Calibration techniques (CAL LOG-LOSS) generally do not significantly impact TabPFN LOG-LOSS's performance for balanced datasets but enhance AutoGluon LOG-LOSS's performance in vulnerable datasets, indicating AutoGluon's poor calibration and TabPFN's well-calibrated class probabilities. However, these techniques can sometimes worsen AutoGluon LOG-LOSS's performance in non-vulnerable systems, leading to MI overestimation and potential false positives in ILD, as shown in Appendix B.2.

3.3.2. Generalization performance

This section examines the generalization capabilities of different MI estimation methods relative to baselines, identifying the best-performing methods using AutoGluon and TabPFN on balanced (non-vulnerable systems with noise levels of $\{0.90, 1.0\}$ and vulnerable systems with noise levels 0.0 and 0.5), and imbalanced binary-class and multi-class datasets containing datasets generated by various vulnerable and non-vulnerable systems using NMAE, as detailed in Section 3.3.1. We assess generalization based on the number of classes (C) and input dimensions (d) by aggregating NMAE across dimensions for each class ($C \in [2, 10]$) and classes for each dimension ($d \in [2, 20]$). For class imbalances, we aggregate NMAE across noise levels for each imbalance parameter ($r \in [0.05, 0.5]$ for binary-class and $r \in [0.02, 0.2]$ for multi-class datasets). Similarly, we assess generalization for noise levels ($\epsilon \in [0.0, 1.0]$) by aggregating NMAE across class imbalances. Figs. 6 and 7 visualize these results with line plots, showing NMAE on systems simulated using MVN perturbation and proximity techniques, respectively.

Number of classes (C) The top row of Figs. 6 and 7 shows MI estimation methods' generalization across datasets with varying numbers of classes ($C \in [2, 10]$) on the X-axis in systems simulated using MVN perturbation and proximity techniques, respectively. In *vulnerable* and *non-vulnerable* systems, AutoGluon and TabPFN show high accuracy (lower NMAE) with increasing classes, with TabPFN outperforming AutoGluon, especially at 50 % noise ($\epsilon = 0.5$). Baselines degrade (NMAE increases) with the increase in the number of classes. In *vulnerable* systems, MINE is the least accurate, whereas in *non-vulnerable* systems, both MINE and PC-SOFTMAX outperform GMM.

Input dimensions (d) The second row of Figs. 6 and 7 shows MI estimation methods' generalization across datasets with varying input dimensions ($d \in [2, 20]$) on the X-axis in systems simulated MVN perturbation and MVN proximity techniques, respectively. In *vulnerable* and *non-vulnerable* systems, AutoGluon and TabPFN improve notably with more dimensions, with TabPFN leading, specifically in systems with 50 % noise. Baselines, especially GMM, deteriorate significantly with more dimensions, indicating that even deep multi-layer perceptrons (MLPs) struggle with high-dimensional datasets. MINE and PC-SOFTMAX perform well in *non-vulnerable* systems simulated using perturbation but deteriorate in ones using the proximity technique.

Class imbalance (r) In the third row of Figs. 6 and 7 shows MI estimation methods' generalization across datasets with varying class imbalances ($r \in [0.05, 0.5]$ for binary-class and $r \in [0.02, 0.2]$ for multi-class) on X-axis, in imbalanced datasets generated using MVN perturbation and MVN proximity technique, respectively. In systems generating imbalanced datasets, TabPFN and GMM show high accuracy and stay unaffected, especially in systems generated using perturbation techniques. AutoGluon, MINE, and PC-SOFTMAX are deteriorating, notably beyond 0.25 *class imbalance* level for systems generating *binary class imbalanced* datasets using perturbation techniques. While for systems generating *imbalanced multi-class* datasets, AutoGluon improves as class imbalance decreases, MINE and PC-SOFTMAX remain relatively constant with a notable drop at 0.04 *class imbalance* level.

Noise level (ϵ) The last row of Figs. 6 and 7 shows MI estimation methods' generalization across datasets with varying noise levels ($\epsilon \in [0.0, 1.0]$) on the X-axis, in imbalanced datasets generated using MVN perturbation and MVN proximity techniques, respectively. In systems generating imbalanced datasets, TabPFN and GMM demonstrate high accuracy across all noise levels, particularly in systems generated using the perturbation technique. AutoGluon deteriorates with increasing noise, while MINE and PC-SOFTMAX improve significantly.

Summary TabPFN, particularly IR CAL LOG-LOSS, consistently demonstrates remarkable generalization in MI estimation across various factors for systems simulated using both MVN perturbation and proximity techniques. TabPFN and AutoGluon show strong generalization performance across different numbers of classes (C) and input dimensions (d), with baselines performing worse as

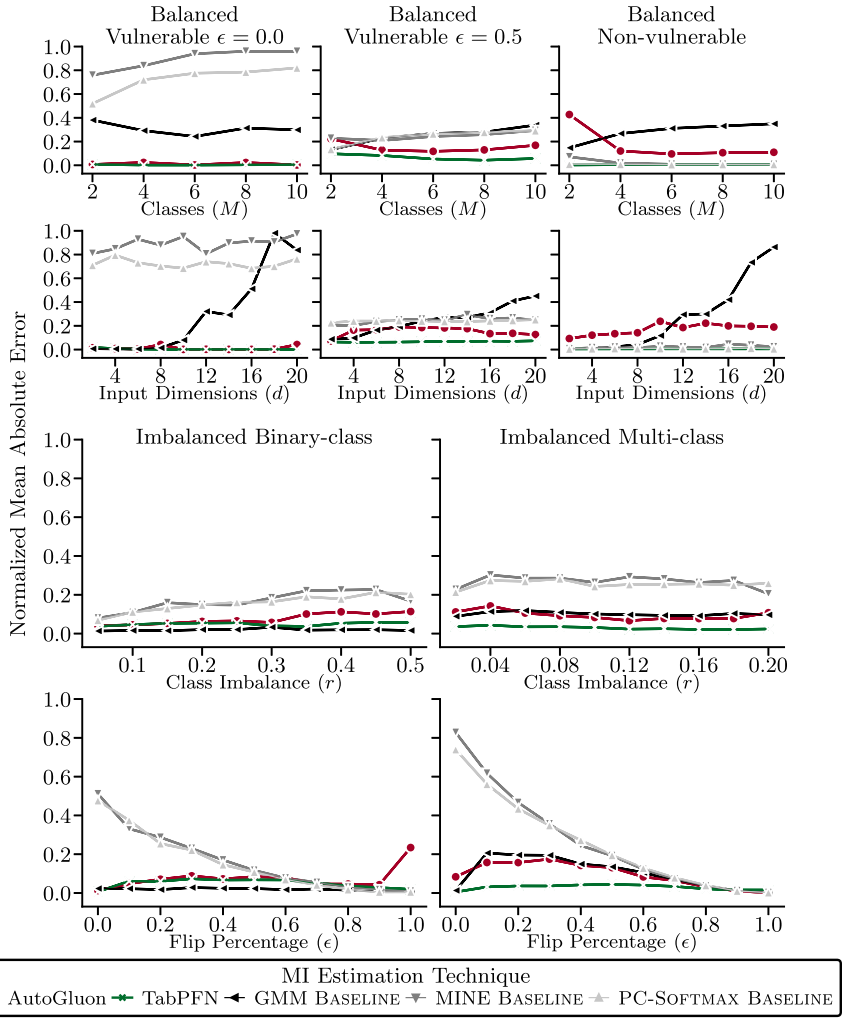


Fig. 6. Generalizability of MI estimation methods on MVN perturbation systems.

they increase. GMM struggles with high-dimensional datasets. MINE shows the lowest generalization capability, suggesting its unsuitability for estimating MI in classification datasets. TabPFN and GMM exhibit robust generalization across various class imbalances (r) and noise levels (ϵ). GMM handles noise and imbalance well in low-dimensional datasets ($d = 5$) but struggles with high-dimensional ones. Conversely, AutoGluon, MINE, and PC-Softmax shows weaker generalization with respect to class imbalances and noise levels.

4. Information leakage detection

This section introduces the ILD process with various classification-based and MI-based ILD approaches. We describe the experimental setup and IL-Datasets descriptions used to evaluate information leakage detection approaches for detecting side-channel leakages (ILs) through time delays, countering Bleichenbacher's attacks on OpenSSL TLS servers, and summarize the results with Appendix B provides a detailed analysis. Our results conclude that our MI-based ILD approach using the calibrated log-loss (Cal Log-Loss) outperforms state-of-the-art methods.

4.1. ILD methodology

Fig. 8 illustrates the process of detecting IL in systems generating classification datasets using various ILD approaches. To enhance IL detection confidence, we apply Holm-Bonferroni correction on p-values obtained using statistical tests on performance estimates of the top-10 AutoML models or pipelines from AutoGluon and TabPFN obtained through HPO.

4.1.1. ILD approaches

We divide the ILD approaches based on using MI and classification metrics to detect IL in a system into MI-based and classification-based methods. We propose using the AutoML tools AutoGluon and TabPFN, as described in Appendix A.2.2, to accurately estimate

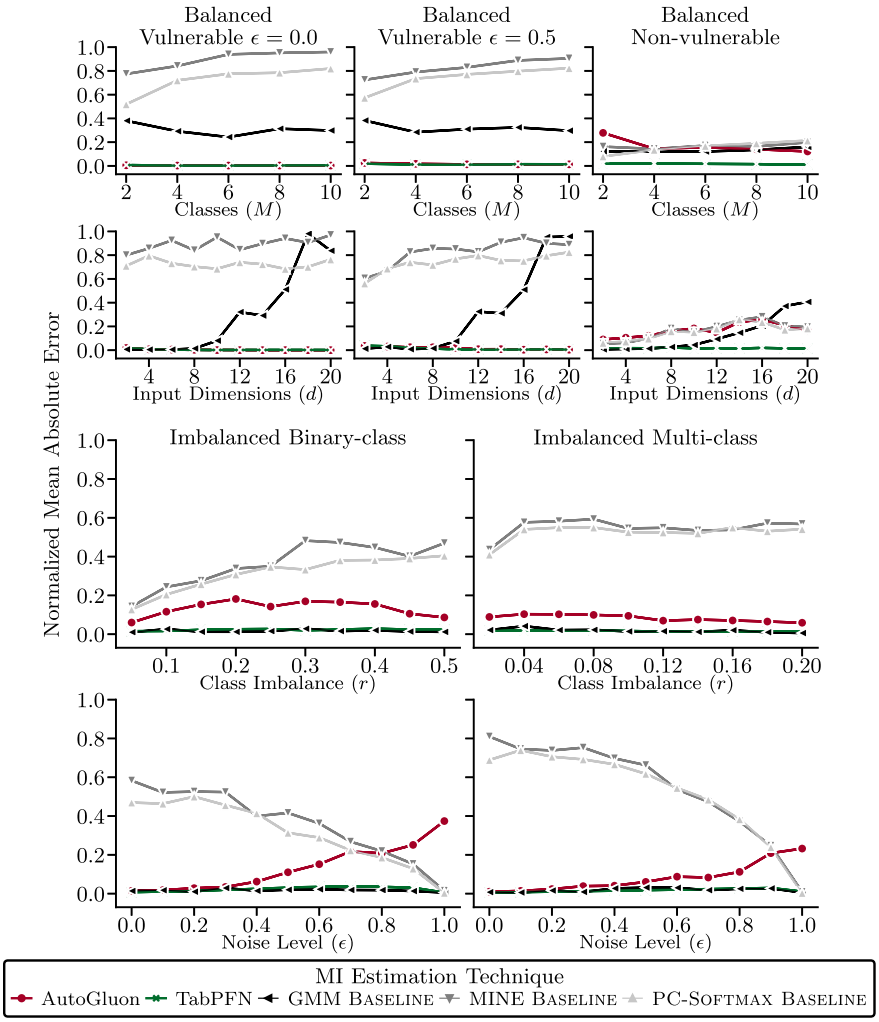


Fig. 7. Generalizability of MI estimation methods on MVN proximity systems.

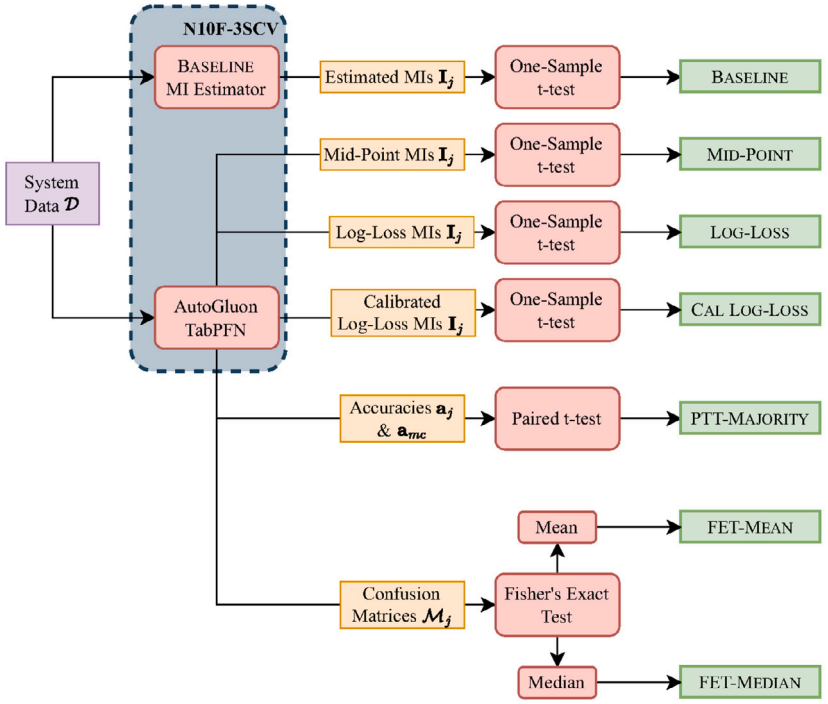
empirical risk minimizers (g or g_p), which can serve as an appropriate proxy for Bayes predictor. We propose applying statistical tests to the LAS calculated using Log-Loss, accuracy, and confusion matrix (CM) of the top-performing AutoML models or pipelines (g or g_p) to obtain a p-value. All statistical tests in our setup follow a common framework, where the null hypothesis $H_0(\cdot)$ assumes no IL, and the alternative $H_1(\cdot)$ indicates its presence. A p-value below the significance level α leads to rejecting H_0, thereby inferring the presence of IL. The threshold $\alpha = 0.01$ is particularly appropriate for avoiding false positives using the Holm-Bonferroni correction when testing multiple hypotheses to ensure high-confidence detection in security-sensitive settings [14,29]. We propose to use $\alpha = 0.01$ for the Holm-Bonferroni correction, tuned with the rejection threshold of 5 on the cutoff parameter $\tau \geq 5$ from our prior work [14].

MI-based approach The MI-based ILD approach is based on the condition that IL occurs in the system if MI or LAS with log-loss must be significantly greater than 0, i.e., $\delta(\ell_{ll}) \gg 0$, as per Section 2.3. To achieve this, we propose to apply the one-sample t-test (OTT) on 10 MI estimates obtained using K-fold cross-validation (KFCV) on the Log-Loss and Mid-Point techniques on top-jth performing pipeline, as well as baseline methods, denoted by vector $\hat{\boldsymbol{I}}_j$. The OTT provides a p-value representing the probability of observing the sample mean to be around the actual mean (0 in our case) [30]. The null hypothesis $H_0(\hat{\boldsymbol{I}}_j \sim 0)$ implies absence of IL in a system.

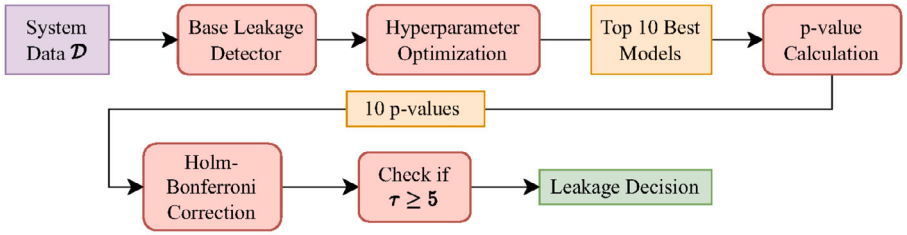
Classification-based approaches Our prior work Gupta et al. [14] proposes classification-based ILD approaches based on the condition that IL occurs if the Bayes predictor accuracy significantly surpasses that of a marginal Bayes predictor, i.e., LAS is significantly greater than 0, as per Section 2.3.

PTT-based approach PTT is used to compare two samples (generated from an underlying population) where the observations in one sample can be paired with observations in the other sample [30]. The approach tests the condition that LAS is significantly greater than 0 for IL to exist in a system, i.e., $\delta(\mathrm{m}_{\mathrm{ACC}}) \gg 0$. The PTT-Majority approach achieves this by applying PTT between the 10 accuracy estimates from KFCV for the j-th best performing AutoML pipeline (g minimizing (5), proxy of Bayes predictor) and

of g_p^{mc} (proxy of marginal Bayes predictor, cf. Section 3.1.2), denoted by $\boldsymbol{a}_j$ and $\boldsymbol{a}_{mc}$, respectively. We propose using the corrected version of PTT, which accounts for the dependency in estimates due to KFCV, to provide an unbiased estimate of the p-value [31]. The p-value represents the probability of obtaining our observed mean accuracy difference, assuming the null hypothesis $H_0(\cdot)$ holds, implying that accuracies are drawn from the same distribution or have nearly zero average difference [30]. The null hypothesis is $H_0(\boldsymbol{a}_j = \boldsymbol{a}_{mc})$, which implies no IL and the alternate hypothesis $H_1(\boldsymbol{a}_j \neq \boldsymbol{a}_{mc})$ implies presence of IL in the system. However, PTT assumes asymptotic behavior and normal distribution of accuracy differences, which can lead to optimistic p-values, and using accuracy can misestimate results on imbalanced datasets [13].



(a) Calculation of p-value by different ILD approaches



(b) Information leakage (IL) detection process

Fig. 8. Procedure of using ILD approaches to detect ILs in a system generating D.

FET-based approach FET is a non-parametric test used to determine the probability of independence (or non-dependence) between two classification methods, in this case, classifying instances based on ground truths $\boldsymbol{y}$ and classifying them based on AutoML predictions $\hat{\boldsymbol{y}}$ [32]. The FET-based approach addresses the class imbalance issue and improves p-value estimation by applying the FET on K CMs $\mathcal{M}_j = \{\boldsymbol{M}_j^k\}_{k=1}^K$ obtained using KFCV to detect IL. If a strong correlation exists between inputs $\boldsymbol{x}$ and outputs y in $\mathcal{D}$, the AutoML pipeline's prediction $\hat{y} = g(\boldsymbol{x})$ encapsulates input information, and FET can assess IL using the CM (analyzing LAS $\delta(\text{m}_{\text{CM}})$), which contains relevant information for predicting correct outputs ($\text{m}_{\text{TP}}, \text{m}_{\text{TN}}$). The p-value represents the probability of independence between the ground truth $\boldsymbol{y}$ and AutoML predictions $\hat{\boldsymbol{y}}$ defined by the null hypothesis ($H_0(\cdot)$). The null hypothesis $H_0(p(\boldsymbol{y}, \hat{\boldsymbol{y}}|\boldsymbol{M}) = p(\boldsymbol{y}|\boldsymbol{M})p(\hat{\boldsymbol{y}}|\boldsymbol{M}))$ posits that $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are independent, indicating no IL, while the alternative hypothesis $H_1(p(\boldsymbol{y}, \hat{\boldsymbol{y}} \mid \boldsymbol{M}) \neq p(\boldsymbol{y} \mid \boldsymbol{M})p(\hat{\boldsymbol{y}} \mid \boldsymbol{M}))$ suggests significant dependence, implying IL.

The FET implicitly accounts for dataset imbalance by testing LAS using Matthews correlation coefficient (MCC) (i.e., $\delta(\text{m}_{\text{MCC}}) \gg 0$). The MCC metric, which equally penalizes m_{FP} and m_{FN} [33], is directly proportional to the square root of the χ^2 statistic, i.e.,

$|\mathrm{m_{MCC}}| = \sqrt{\chi^2/N}$ [32]. Since the χ^2 test is asymptotically equivalent to FET, this supports using FET to evaluate the learnability of g (a proxy for the Bayes predictor) via the CM, while accounting for class imbalance and yielding accurate p-values for detecting IL. We aggregate the 10 p-values obtained using FET on K CMs through *median* and *mean*, referred to as the FET-MEDIAN and FET-MEAN approaches [34].

4.1.2. Detection process

The ILD process involves rigorous estimation, statistical assessment, and correction techniques to detect IL in a system confidently. The process starts with obtaining accurate MI or Bayes predictor performance estimates using nested K-fold cross-validation (KFCV) with HPO and specific parameter ranges provided in Table A.5. The dataset $\mathcal{D}$ is split into 90 % training and 10 % test sets using KFCV (10), conducting HPO with 100 evaluations using MCCV with 3 splits, reserving 30 % of training data for validation, denoted as "N10F-3SCV", as depicted in Fig. 8b Objective functions for HPO include BER for PC-SOFTMAX, AutoGluon, and TabPFN, AIC for GMM, and MSE for MINE with parameter ranges provided in Table A.5. We identify the top-10 best-performing pipelines from running AutoGluon for 1800 seconds and top-performing models using HPO on GMM, MINE PC-SOFTMAX and TabPFN, using validation loss or accuracy. KFCV is applied to the top-10 models or pipelines, generating 10 estimates from the entire dataset $\mathcal{D}$, which statistical tests use to produce p-values, which are assessed to detect IL. We obtain 10 MI estimates, accuracies, and CMs for each of the j-th best-performing model/AutoML pipeline, denoted by $\hat{\boldsymbol{I}}_j$, $\boldsymbol{a}_j$ and $\mathcal{M}_j = \{\boldsymbol{M}_j^k\}_{k=1}^K$, respectively. The OTT is used on MI estimates ($\hat{\boldsymbol{I}}_j$), PTT compares the accuracies ($\boldsymbol{a}_j$) with that of g_p^{mc} ($\boldsymbol{a}_{mc}$), and FET is applied to CMs ($\mathcal{M}_j = \{\boldsymbol{M}_j^k\}_{k=1}^K$), producing 10 p-values which are aggregated using *mean* and *median* operators.

To enhance **robustness and reliability** in ILD, we use the Holm-Bonferroni correction to ensure accuracy and reliability in our ILD framework, which mitigates the influence of overfitting and noisy estimates that can arise when using just one pipeline (model) [29]. The Holm–Bonferroni method controls the family-wise error rate, minimizing false positives (type-1 errors) by adjusting the rejection criteria α for each hypothesis within our family of null hypotheses, $\mathcal{F} = \{H_1, \ldots, H_J\}$, ensuring that the significance level of $\mathcal{F}$ not exceeding predefined threshold of $\alpha = 0.01$ [29]. After obtaining 10 p-values from top-10 models/pipelines, we apply the correction to acquire the number of rejected hypotheses or the cut-off parameter $\tau = |\mathcal{F}_r|$, quantifying IL detection confidence. To detect IL efficiently, it is imperative to set an appropriate **rejection threshold** on the cut-off parameter τ. A higher rejection threshold would help avoid false positives and prevent the detection of non-existent IL while decreasing it would avoid missing ILs occurrences and reduce false negatives. Based on prior work [14], setting the rejection threshold to $\lfloor J/2 \rfloor$ on the cut-off parameter τ ensures robust and accurate IL detection, which we also use for this study, i.e., $\tau \geq 5$. This systematic and rigorous approach detects IL with a high degree of confidence, ensuring the robustness of our ILD framework.

4.2. Empirical evaluation

This section outlines the evaluation process for our ILD approaches compared to baselines in detecting timing side-channel leaks in OpenSSL TLS servers, as illustrated in Fig. 8. Our main objective is to assess the generalization capability of various ILD approaches with respect to the LAS of systems, generating both balanced and imbalanced datasets, as outlined in Table 2. The overall performance results for selected approaches are discussed in Section 4.3, with detailed analysis in Appendix B.

4.2.1. OpenSSL timing datasets

We target side-channel vulnerabilities in cryptographic software to validate our approaches using network traffic generated by a modified OpenSSL TLS server. Bleichenbacher's attack exploits server behavior to differentiate correctly formatted decrypted messages from incorrect ones [35]. A secure OpenSSL TLS server exhibits no time difference in processing correctly and incorrectly formatted messages, but a vulnerable server does, exposing IL through processing time differences. For our experiments, the timing side-channel or time delay, i.e., observable differences in server computation times when processing messages with correct and manipulated padding were introduced as per the Java TLS implementation vulnerability `CVE-2014-0411`[1] [36]. Datasets were provided by Funke [37] from the vulnerable (DamnVulnerableOpenSSL[2]) OpenSSL TLS server, which exhibits longer computation times for incorrectly formatted messages with manipulated padding, and the non-vulnerable (OpenSSL 1.0.2l[3]), which shows no time delay (no IL). There are 10 padding manipulations: five cause longer processing times (simulating IL, $z = 1$), and five do not (no IL, $z = 0$). Each IL-Dataset $\mathcal{L}$ contains 10 binary-class datasets corresponding to 10 padding manipulations $\mathcal{D}$, with label 0 instances corresponding to correctly formatted messages and positive label ($y = 1$) instances corresponding to incorrectly formatted messages. Table 2 details the IL-Datasets generated for various time delay values (in μ seconds) serving as the system's LAS, and class imbalances of $r \in \{0.1, 0.3, 0.5\}$ uploaded on OpenML.[4]

4.2.2. Experimental setup

We apply each ILD approach depicted in Fig. 8b to a set of IL-Datasets detailed in Table 2. As discussed IL-Dataset ($\mathcal{L} = \{(\mathcal{D}_i, z_i)\}_{i=1}^{10}$) consists of 10 systems corresponding to 10 messages with manipulated padding ($y_i = 1$), with five systems containing IL ($z = 1$) and five

[1] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0411.
[2] https://github.com/tls-attacker/DamnVulnerableOpenSSL.
[3] https://www.openssl.org/source/old/1.0.2/.
[4] https://www.openml.org/search?type=study&sort=tasks_included&study_type=task&id=383.

Table 2
Overview of the OpenSSL timing IL-Datasets used for the ILD experiments.

Time Delay (in μ seconds)	# Folds	Imbalance r	IL-Dataset $\mathcal{L}$ configuration			Dataset D configuration			
			# Systems $\lvert\mathcal{L}\rvert$	# $z=0$	# $z=1$	$\lvert D\rvert$	# $y=0$	# $y=1$	# Features
$\{2^0, 2^1, \ldots, 2^8\}$	3	0.1	10	5	5	[943, 1064]	[849, 958]	[94, 106]	124
$\{2^0, 2^1, \ldots, 2^8\}$	3	0.3	10	5	5	[1212, 1368]	[849, 958]	[363, 410]	124
$\{2^0, 2^1, \ldots, 2^8\}$	3	0.5	10	5	5	[1725, 1929]	[849, 958]	[829, 989]	124
{10, 15, …, 35}	10	0.1	10	5	5	[962, 2263]	[866, 2037]	[96, 226]	[124, 154]
{10, 15, …, 35}	10	0.3	10	5	5	[1237, 2910]	[866, 2037]	[371, 873]	[124, 154]
{10, 15, …, 35}	10	0.5	10	5	5	[1721, 4084]	[866, 2037]	[826, 2104]	[124, 154]

not ($z=0$). We evaluate each ILD approach using standard binary classification metrics: Accuracy, FPR, and FNR (cf. Appendix A.2.1), on the predicted IL decisions ($\hat{\boldsymbol{z}} = (\hat{z}_1, \ldots, \hat{z}_{10})$) and the ground truth vector ($\boldsymbol{z} = (z_1, \ldots, z_{10})$).

4.3. Results: detection accuracy

We evaluate the performance of various ILD approaches across different class imbalance levels ($r \in \{0.1, 0.3, 0.5\}$) and time delays. Our analysis distinguishes between OpenSSL systems with short (≤ 25 μ-seconds) and long (≥ 25 μ-seconds) delays, reflecting complex and straightforward IL cases, respectively. Fig. 9 shows the mean detection accuracy and SE across IL-Datasets using a rejection threshold of 5 on τ. For generalization results across varying LAS or time delays, see Appendix B.

We summarize the results by selecting the *best-performing calibrated LOG-LOSS ILD approaches* using both TabPFN and AutoGluon using LOG-LOSS estimation, we assessed their NMAE from experiments detailed in Section 3.2.2. This process determined the optimal calibration approach for each AutoML tool separately for balanced ($r = 0.5$) and imbalanced ($r \in 0.1, 0.3$) datasets. AutoGluon TS CAL LOG-LOSS was most effective for both scenarios, while TabPFN IR CAL LOG-LOSS performed best on imbalanced datasets and TabPFN PS CAL LOG-LOSS for balanced ones. We focus on FET-MEDIAN due to its robustness [34], and prior work [14] shows it performs similarly to FET-MEAN at a rejection threshold of 5 for the cut-off parameter τ.

4.3.1. Short time delay

Our first set of experiments focuses on systems with short time delays (≤ 25 μ-seconds) with more complex or noisy ILs.

Imbalanced Detecting ILs with short time delays or LAS is challenging. Notably, the TabPFN CAL LOG-LOSS approach consistently exhibits high detection accuracy, with AutoGluon FET-MEDIAN also performing well. For $r = 0.1$, TabPFN CAL LOG-LOSS achieves 69 % accuracy and 58 % for $r = 0.3$, while AutoGluon FET-MEDIAN detects 54 % and 58 % of ILs, respectively. However, detecting ILs in these scenarios remains challenging due to missed IL and high false positive rate (FPR), as detailed in Appendix B.

Balanced Detecting ILs in systems generating balanced datasets with short time delays remains challenging, but it is more manageable than generating imbalanced ones. AutoGluon LOG-LOSS and CAL LOG-LOSS approaches outperform, detecting a significant proportion of ILs (approximately 68 %). IR CAL LOG-LOSS enhances TabPFN LOG-LOSS detection accuracy to 67 %, making it a competent approach, while PS CAL LOG-LOSS does not improve its detection performance. Overall, AutoGluon outperforms TabPFN, with PTT-MAJORITY detecting over 61 % of ILs.

4.3.2. Long time delay

Our second set of experiments focuses on systems with long time delay (≥ 25 μ-seconds) with easily detectable ILs.

Imbalanced Detecting ILs in systems with larger time delays or LAS is more straightforward, with all approaches consistently performing well, detecting more than 50 % of ILs. In particular, the TabPFN CAL LOG-LOSS approach detects approximately 94 % of ILs, while AutoGluon FET-MEDIAN excels, detecting over 92 % of ILs for $r = 0.3$ and approximately 72 % for $r = 0.1$. FET based approaches perform better than PTT-MAJORITY in detecting ILs in imbalanced systems' datasets, also confirmed in [14].

Balanced In systems with balanced datasets and longer time delays, detecting ILs becomes more straightforward. AutoGluon MID-POINT and TabPFN IR CAL LOG-LOSS consistently detect the majority of ILs (around 99 %), confirming that IR CAL LOG-LOSS provides an effective alternative calibration technique to enhance the performance of the TabPFN LOG-LOSS approach. In contrast, TabPFN PS CAL LOG-LOSS and LOG-LOSS detect approximately 97 % of ILs, showing no improvement with the selected calibration method, while TabPFN MID-POINT detects only 85 %. Among classification-based approaches, PTT-MAJORITY consistently outperforms others, detecting 99 % of ILs, which contrasts with the findings in [14].

4.3.3. Summary

Detecting IL is easier in systems that generate balanced datasets, where TabPFN IR CAL LOG-LOSS, MID-POINT, and PTT-MAJORITY, using AutoGluon, show robust performance. Contrary to Gupta et al. [14], PTT-MAJORITY outperforms FET-based approaches in balanced datasets, indicating that ILD performance varies with IL patterns. Baseline methods detect only 50 % of ILs, reflecting random

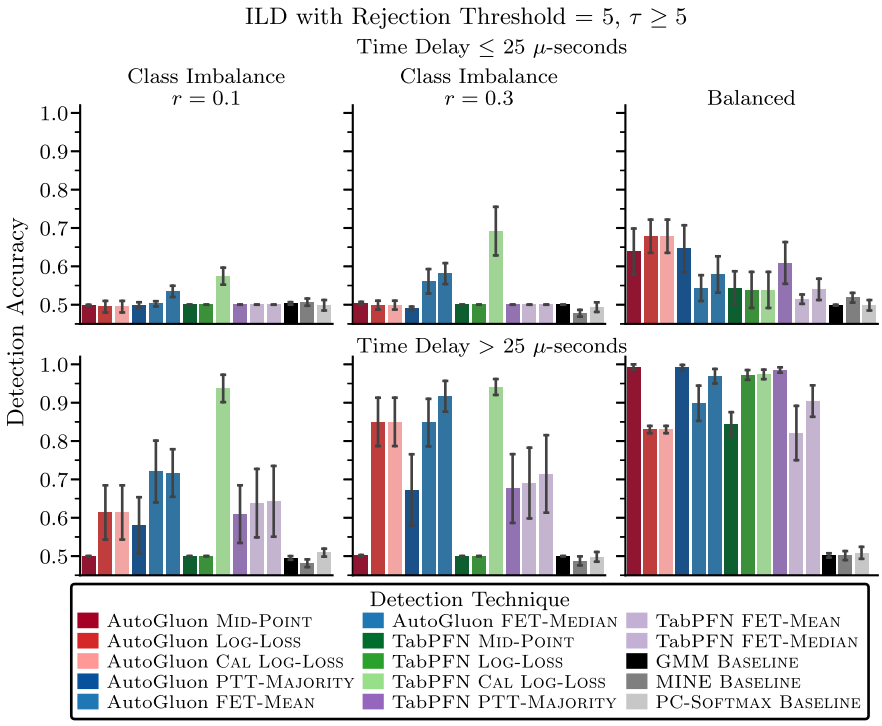


Fig. 9. ILD performance in detecting OpenSSL TLS timing side-channels.

Table 3
Runtime comparison (in seconds) for each base learner for all experiments.

Learner	ILD Experiments	MI Estimation Experiments
AutoGluon	2395.55 ±5.21	7168.04 ±62.77
TabPFN	30513.28 ±114.85	9970.27 ±72.34
GMM Baseline	35113.05 ±348.72	963.79 ±22.36
MINE Baseline	33447.14 ±128.25	1962.87 ±13.43
PC-softmax Baseline	16993.88 ±266.83	7605.09 ±91.36
Total	18027.93 ±148.12	7676.46 ±42.60

detection decisions, as discussed in Appendix B. TabPFN Cal Log-Loss consistently outperforms other approaches, highlighting the need for calibrating TabPFN Log-Loss for accurate MI estimation. The Mid-Point approach is unable to detect ILs in systems generating imbalanced datasets, as expected. For balanced datasets, IR Cal Log-Loss significantly improves detection accuracy compared to PS Cal Log-Loss, as shown in Appendix B, indicating that the choice of calibration technique depends on the available underlying dataset. Calibration of Log-Loss using AutoGluon often leads to overfitting and overestimating MI, resulting in false positives, as also shown in Section 3.3.1. This discrepancy may result from feature reduction (from 150 to [20, 50]) on an imbalanced dataset using TabPFN, implying that the choice and necessity of calibration techniques depend on the underlying datasets.

4.3.4. Computational time

For practical use, AutoGluon completes an ILD run in approximately 2395 seconds, while TabPFN requires around 30513 seconds, as detailed in Table 3. Although the transformer-based inference offers more accurate MI estimation, making it more suitable when detection accuracy is prioritized over runtime, as shown in Fig. 9. TabPFN's ability to produce high-quality predictions through a single forward pass reduces the need for calibration in MI estimation. However, calibration may still be necessary depending on the use of feature reduction and the presence of class imbalance in the dataset. The baselines are computationally more expensive than AutoGluon and perform poorly in detecting ILs, further indicating that AutoGluon offers a favorable trade-off between efficiency and accuracy, making it a computationally effective choice for practical ILD deployment. With TabPFN now integrated into AutoGluon and capable of handling datasets with over 100 features, it is well-suited for inclusion in a fully automated pipeline that supports appropriate calibration techniques for estimating MI via Log-Loss and enables scalable ILD in high-dimensional settings [38].

5. Conclusion

This paper presents a comprehensive framework for ILD, leveraging information theory and statistical learning theory concepts to enhance cybersecurity by identifying vulnerabilities. We introduced two techniques for quantifying and detecting IL by estimating MI between a system's observable and secret information, using the Bayes predictor's Log-Loss and accuracy. Our methods employ two powerful AutoML tools, TabPFN and AutoGluon, for efficient MI estimation, offering an automated alternative to traditional statistical techniques or deep MLPs. We propose to apply OTT to estimated MIs and enhance robustness and provide confidence in IL decisions using Holm-Bonferroni correction, as already established in our prior work [14].

Our empirical results show that our approach, mainly using TabPFN for calibrated Log-Loss approximation, effectively estimates MI and detects timing side-channel leaks in OpenSSL TLS servers, outperforming state-of-the-art methods. The key contributions of our work include a comprehensive ILD framework, addressing imbalance with minimal false positives, robustness to noise in generated system datasets, and an adaptable, scalable IL detection solution for real-world scenarios. Furthermore, our work concludes that the choice and requirement of calibration techniques for Log-Loss estimation depend on the characteristics of the system datasets.

In future work, we aim to use attention-based transformer embeddings to reduce high-dimensional network traces into compact, semantically meaningful representations, enabling the effective quantification and detection of IL while addressing TabPFN's input limitations. The new version of AutoGluon supports advanced search algorithms such as Hyperband, which can be explored to accelerate the overall process of finding an optimal model using evaluation-efficient successive halving [39]. As TabPFN is now part of AutoGluon's model pool [38], a promising direction is to integrate it with feature selection and dimensionality reduction for building an end-to-end AutoML pipeline. This would enable scalable MI estimation and IL detection on high-dimensional, multi-class datasets ($C > 10$), thereby overcoming the limitations of the proposed solution. Improving the Mid-Point approach to account for dataset imbalance and extending the BER and MI relationship by Zhao et al. [25] for multiple classes is an essential direction for future work as well. We plan to extend our experiments to finer-grained imbalance and time-delay variations for deeper robustness analysis. Another direction involves integrating permutation-based tests, such as Westfall–Young, and tuning the Holm-Bonferroni rejection threshold to the standard significance level of $\alpha = 0.05$ [30].

Additionally, we will explore detecting leaks through side information leaked via channels such as the CPU caches, power consumption, and electromagnetic radiation [7,9]. [40] showed that the MSE of the Bayes-optimal predictor in regression tasks, where the output is real-valued, is directly linked to the MI between input and output. This relationship enables the use of MI to quantify and detect leakage in systems vulnerable to cloud-based SCAs, where continuous-valued user information, such as age or salary, may be exposed [41]. Exploring such leakage scenarios represents a promising direction for future work [42]. We aim to explore adaptive IL detection methods for evolving attack strategies or environments, using approaches such as reinforcement or active learning [43].

CRediT authorship contribution statement

Pritha Gupta: Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization. **Marcel Wever:** Writing – review & editing, Supervision, Investigation. **Eyke Hüllermeier:** Writing – review & editing, Supervision, Formal analysis.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pritha Gupta reports a relationship with the Software Innovation Campus Paderborn (SICP) at Paderborn University's Department of Computer Science. The remaining co-authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We are grateful to Dennis Funke, Jan Peter Drees, Karlson Pfannschmidt, Arunselvan Ramaswamy, Björn Haddenhorst, Stefan Heid, and Shambhavi Gupta for their valuable suggestions. We acknowledge computing time on the high-performance computers Noctua2 at the NHR Center PC2 under the project AProSys, Paderborn University for performing the simulations, granted under the project "hpc-prf-aiafs" (AProSys) project, Förderkennzeichen: 03EI6090E under Abrechnungsobjekt: 3130500154. We also acknowledge the funding provided by AutoSCA, Paderborn University project, supported by Bundesministerium für Bildung und Forschung (BMBF) with Förderkennzeichen: 16KIS1190 and European Research Council (ERC) with 802823.

Appendix A. Additional experimental details

This section will list all experimental details excluded from the main paper for conciseness. Hyperparameters and their ranges for TabPFN, AutoGluon, baseline MI estimators, and ILD approaches are provided in Table A.5. We provide a detailed explanation of the algorithms used to generate synthetic system datasets, a description of the performance metrics, and the AutoML tools employed in the experiments implemented in the Python package,[5] along with the scripts used for running the experiments in Sections 3 and 4.2

[5] https://github.com/LeakDetectAI/AutoMLQuantILDetect.

Table A.4
Notation used throughout the paper.

Symbol	Meaning
$[n]$	Set of integers $\{1, 2, \ldots, n\}, n \in \mathbb{N}$
$[a, b]$	Set of integers $\{a, a+1, \ldots, b\}$, $a, b \in \mathbb{N}$
$[\![A]\!]$	Indicator function which is 1 if statement A is true and 0 otherwise
$p_{(X,Y)}(\cdot)$, $\hat{p}_{(X,Y)}(\cdot)$	Actual and predicted joint PDF between (X, Y)
$p_{(X,Y)}(\boldsymbol{x}, y) = p_{(X,Y)}(\boldsymbol{x}, y)$	Joint PDF of X and Y, at point $(\boldsymbol{x}, y)$
$p_X(\cdot)$, $\hat{p}_X(\cdot)$	Actual and predicted marginal PDF of X
$p_X(\boldsymbol{x}) = p_X(\boldsymbol{x})$	Probability mass of input $\boldsymbol{x}$
$p_Y(\cdot)$, $\hat{p}_Y(\cdot)$	Actual and predicted marginal PMF of Y
$p_Y(y) = p_Y(y)$	Probability of class label y
$p_{Y\|X}(\cdot)$, $\hat{p}_{Y\|X}(\cdot)$	Actual and predicted conditional PDF of Y given X
$p_{Y\|X}(y \mid \boldsymbol{x}) = p_{Y\|X}(y \mid \boldsymbol{x})$	Probability of y given $\boldsymbol{x}$
$p_{X\|Y}(\cdot)$, $\hat{p}_{X\|Y}(\cdot)$	Actual and predicted conditional PDF of Y given X
$p_{X\|Y}(\boldsymbol{x} \mid y) = p_{X\|Y}(\boldsymbol{x} \mid y)$	Probability of $\boldsymbol{x}$ given y
X	Input ($\boldsymbol{x} \in \mathbb{R}^d$) random variable (d-dimensional continuous)
Y	Output ($y \in [C]$) random variable (discrete)
$\mathcal{X} \in \mathbb{R}^d$	Input Space, set of $\boldsymbol{x}$ sampled from X
$\mathcal{Y} \in [C]$	Output Space, set of y sampled from Y
$D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$	Classification dataset
r	Imbalance in a dataset D
ϵ	Noise in a dataset D
$I(X;Y)$	MI between X and Y
$H(Y \mid X)$	Conditional entropy for Y given X
$H(X), H(Y)$	Entropy for random variable X and Y
$H_2(a) = -(a)\log(a) - (1-a)\log(1-a)$	Binary cross-entropy function for $a \in (0, 1)$
$\mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{bc}}), \mathrm{m}_{\mathrm{ERR}}(g^{\mathrm{mc}})$	The Bayes error rate and error rate of marginal Bayes predictor
$\delta(\ell_{(\cdot)}) = \ell_{(\cdot)}(g^{\mathrm{mc}}) - \ell_{(\cdot)}(g^{\mathrm{bc}})$ $\delta(\mathrm{m}_{(\cdot)}) = \lvert\mathrm{m}_{(\cdot)}(g^{\mathrm{mc}}) - \mathrm{m}_{(\cdot)}(g^{\mathrm{bc}})\rvert$	LAS is the difference in performance of g^{bc} and g^{mc} quantifying IL
$L, \mathcal{L}$	ILD function and IL-Dataset
$H_0(\text{condition}), H_1(\text{condition})$	Null and Alternate hypothesis for statistical tests
$\tau \in [J], J \in \mathbb{N}$,	Cut-off parameter on J hypothesis for Holm-Bonferroni correction
$\alpha = 0.01$	Significance level on H_0 (accept H_1) for statistical tests

and additional results.[6] The 10 IL-Datasets per time delay were transformed into multi-class datasets with 11 padding classes and uploaded to OpenML.[4]

A.1. Simulating synthetic systems

We use the MVN distribution, which is ideal for simulating real-world vulnerable and non-vulnerable systems generating classification datasets, providing a straightforward means of obtaining ground truth MI, as described in Appendix A.1.3. The MVN distribution is widely used for benchmarking classifiers due to its simplicity, ability to model inter-variable correlations, and well-established statistical properties [17, chap. 2]. Its use is further supported by the central limit theorem, which justifies normality in aggregated data [17, chap. 4]. We simulate both balanced and imbalanced datasets with varying LAS or MI levels using MVN perturbation and proximity techniques.

A.1.1. Generation method

Synthetic datasets are generated using MVN to define the joint PDF ($p_{(X,Y)}$) between X and Y, inducing their marginals on X and Y. The dataset D is created by sampling instances from $p_{X|Y}(\cdot)$ with class distribution $p_Y(y)$, as illustrated in Fig. 2.

Formal definition of PDFs We define the joint distribution $p_{(X,Y)}(\cdot)$ and marginal on Y $p_X(\cdot)$, required to generate the dataset D by sampling $(\boldsymbol{x}_i, y_i) \sim p_{X|Y}(\boldsymbol{x}_i \mid y_i), \forall i \in [N]$ with class distribution defined by $p_Y(y)$. These PDFs are used to define the conditionals $p_{Y|X}(\cdot)$ and $p_{X|Y}(\cdot)$ and induce the marginal on $\mathcal{X}$, denoted by $p_X(\cdot)$. The conditional PDF on X given Y is defined for class c as

$$p_{X|Y}(\boldsymbol{x} \mid c) = \mathrm{MVN}(\boldsymbol{\mu}_m, \Sigma), \tag{A.1}$$

where Σ is the covariance matrix and $\boldsymbol{\mu}_m$ is the mean vector for class c. The covariance matrix should be positive semi-definite and is typically sampled using eigenvalue decomposition [17].

Generating imbalanced datasets: marginal on Y The parameter r, referred to as **class imbalance**, is the minimum proportion of instances for any class $c \in [C]$, defined as $r = \min_{c \in [C]} \frac{|\{(\boldsymbol{x}_i, y_i) \in D \mid y_i = c\}|}{|D|}$. For balanced datasets, $r = 1/C$, and for imbalanced ones,

[6] https://github.com/LeakDetectAI/automl-qild-experiments/.

Table A.5
Hyperparameter ranges for AutoML tools: AutoGluon models and TabPFN including the MI estimation baseline approaches (GMM, MINE, and PC-Softmax).

AutoGluon								
Tree-based Ensemble Models								
Base Learner	Learning Rate	# Estimators	Max Depth	# Leaves	Feature Fraction	Bagging Fraction	Min Data in Leaf	Lambda L1 / Lambda L2
Light gradient boosting machine (LightGBM)	[0.01, 0.5]	[20, 300]	[3, 20]	[20, 300]	[0.2, 0.95]	[0.2, 0.95]	[20, 5000]	[1e−6, 1e−2]
Categorical boosting machine (CatBoost)	[0.01, 0.5]	NA	[4, 10]	NA	NA	NA	NA	[0.1, 10]
EXtreme gradient boosting machine (XGBoost)	[0.01, 0.5]	[20, 300]	[3, 10]	NA	NA	NA	NA	NA
Random forest classifier (RF)	NA	[20, 300]	[6, 20]	NA	NA	NA	NA	NA
Extra trees classifier (XT)	NA	[20, 300]	[6, 20]	NA	NA	NA	NA	NA
Neural Networks (MLPs)								
Base Learner	Learning Rate	Dropout Prob	# Layers	# Units	Other Parameters			
FASTAI	[1e−5, 1e−1]	[0.0, 0.5]	NA	NA	NA	NA	NA	NA
NN_TORCH	[1e−5, 1e−1]	[0.0, 0.5]	[2, 20]	[8, 256]	NA	NA	NA	NA
TabPFN								
Base AutoML Model	Reduction Technique	#Reduced Features	# Ensembles	Other Parameters				
TabPFN	RF, XT	[10, 50]	[32, 200]	NA	NA	NA	NA	NA
Baselines								
Learner	Reduction Technique	# Reduced Features	Covariance Matrix Type	Regularization Strength	Other Parameters			
GMM	RF, XT	[10, 50]	{Full, Diagonal, Tied, Spherical}	[1e−10, 1e−1]	NA	NA	NA	NA
	Learning Rate	Optimizer Type	# Layers	# Units	Regularization Strength	Early Stopping	Batch Normalization	Other Parameters
MINE	[1e−5, 1e−1]	{`RMSProp`, `SGD`, `Adam`}	[1, 50]	[2, 256]	[1e−10, 0.2]	{True, False}	{True, False}	NA
PC-Softmax	[1e−5, 1e−1]	{`RMSProp`, `SGD`, `Adam`}	[1, 50]	[2, 256]	[1e−10, 0.2]	{True, False}	{True, False}	NA

$r < 1/C$, i.e., $r \in (0, 1/C]$. To generate imbalanced multi-class datasets ($C > 2$), we introduce two methods: *Minority* and *Majority*, with an example class frequency in each case shown in Fig. 3. In the *Minority* method, the **minority class** is assigned r fraction of total data points, and remaining samples are uniformly distributed among other classes ($> \frac{N}{C}$). While in the *Majority* method, all classes apart from the selected **majority class** is assigned r fraction of total data points ($< \frac{N}{C}$), and the **majority class** gets the remaining data points.

The marginal on Y for a balanced dataset with $r = 1/C$ is defined as $p_Y(c) = 1/C$ and for imbalanced datasets using *Minority* or *Majority* generation methods is defined as

$$\mathrm{g}_r = \begin{cases} \text{Majority} & \begin{cases} p_Y(c) = r, & \text{if } c \in [C] \setminus C \\ p_Y(c) = 1 - r \cdot (C-1), & \text{if } c = C \end{cases} \\ \hline \text{Minority} & \begin{cases} p_Y(c) = \frac{1-r}{C-1}, & \text{if } c \in [C] \setminus C \\ p_Y(c) = r, & \text{if } c = C \end{cases} \end{cases} \tag{A.2}$$

Using these, the joint distribution $p_{(X,Y)}(\boldsymbol{x}, c)$ for class c is defined as $p_{(X,Y)}(\boldsymbol{x}, c) = p_Y(c) \cdot p_{X|Y}(\boldsymbol{x} \mid c)$, which induces a marginal on X as

$$p_X(\boldsymbol{x}) = \sum_{c=1}^{C} p_Y(c) \cdot p_{X|Y}(\boldsymbol{x} \mid c) = \sum_{c=1}^{C} p_Y(c) \cdot \mathrm{MVN}(\mu_m, \Sigma) \tag{A.3}$$

Conditional on Y given X The conditional $p_{Y|X}(\cdot)$ on Y given X for instance $(\boldsymbol{x}, c)$ is defined as

$$p_{Y|X}(c \mid \boldsymbol{x}) = \frac{p_Y(c) \cdot p_{X|Y}(\boldsymbol{x} \mid c)}{\sum_{c=1}^{C} p_Y(c) \cdot p_{X|Y}(\boldsymbol{x} \mid c)} = \frac{p_{(X,Y)}(\boldsymbol{x}, c)}{p_X(\boldsymbol{x})}. \tag{A.4}$$

A.1.2. Introducing noise (ϵ)

To simulate real-world IL scenarios in cryptographic systems, we introduce noise, which decreases the certainty about output $y_i \in \mathcal{Y}$ with more observed inputs $\boldsymbol{x}_i \in \mathcal{X}$, resulting in $H(Y \mid X) > 0$ and $I(X;Y) < H(Y)$. We propose two methods: the MVN perturbation and proximity techniques. The perturbation technique introduces noise by flipping a percentage of outputs or classes in the dataset. In contrast, the proximity technique reduces the distance between mean vectors ($\boldsymbol{\mu}_c$) of each class, leading to overlap between the Gaussians $\mathrm{MVN}(\boldsymbol{\mu}_c, \Sigma)$. These methods simulate scenarios where cryptographic systems leak no information (non-vulnerable) with $I(X;Y) = 0$.

MVN perturbation technique This approach introduces noise by flipping a percentage, denoted by ϵ, of class labels ($y \sim \mathcal{Y}$) in the dataset, simulating perturbed systems to generate classification datasets. This technique modifies the conditional PDFs $p_{Y|X}(\cdot)$ in (A.4) and $p_{X|Y}(\cdot)$ in (A.1).

Output variable Y_ϵ Let random variables $B \sim \mathrm{Bernoulli}(p = \epsilon, q = 1 - \epsilon)$ and $\tilde{Y} \sim \mathrm{Categorical}(p_Y(1), \ldots, p_Y(C))$ are independent of $\{X, Y\}$, for a fixed ϵ. The modified output random variable Y_ϵ is defined as $Y_\epsilon = Y \cdot [\![B = 0]\!] + [\![B = 1]\!] \cdot \tilde{Y}$. The marginal distributions on X remain unchanged because only the class labels are flipped. Accordingly, the modified marginal on Y_ϵ is derived as $p_{Y_\epsilon}(Y_\epsilon = y) = \epsilon \cdot p_Y(Y = y) + (1 - \epsilon) \cdot p_Y(Y = y) = p_Y(Y = y)$. So, the marginal on Y_ϵ is the same as that of Y.

Conditional on Y_ϵ given X The conditional on the modified output variable Y_ϵ given X is

$$p_{Y_\epsilon|X}(Y_\epsilon = c \mid \boldsymbol{x}) = p_B(B = 0) \cdot p_{Y|X}(c \mid \boldsymbol{x}) + p_B(B = 1) \cdot p_{\tilde{Y}}(\tilde{Y} = c) \tag{A.5}$$

$$= (1 - \epsilon) \cdot p_{Y|X}(c \mid \boldsymbol{x}) + \epsilon \cdot p_Y(c). \tag{A.6}$$

Conditional on X given Y_ϵ The conditional $p_{X|Y_\epsilon}(\boldsymbol{x} \mid Y_\epsilon = c)$ combines multiple MVN components is defined as

$$p_{X|Y_\epsilon}(\boldsymbol{x} \mid Y_\epsilon = c) = \frac{p_{Y_\epsilon|X}(Y_\epsilon = c \mid \boldsymbol{x}) p_X(\boldsymbol{x})}{p_Y(c)},.$$

This results in a more complex distribution, which is more difficult to approximate for performing MI estimation.

MVN proximity technique Our second approach introduces noise in the simulated systems by reducing the distance between the Gaussians generated by MVN distributions. This is achieved by moving the mean vectors $\boldsymbol{\mu}'_c, \forall c \in [C]_0$ corresponding to MVN distribution representing each class closer to each other. The updated MVNs for each class c is defined by $\mathrm{MVN}(\boldsymbol{\mu}'_c, \Sigma)$, which in turn represents the conditional PDF $p_{X|Y}(\cdot)$ for the underlying generated system dataset. Notably, when introducing proximity, only the original MVN is modified, which only modifies the conditional on X given Y, i.e., $p'_{X|Y}(\boldsymbol{x} \mid c) = \mathrm{MVN}(\boldsymbol{\mu}'_c, \Sigma)$. Consequently, the conditional PDF $p_{Y|X}(\cdot)$ and the marginal on X can be computed using (A.4) and (A.3) as

$$p'_{Y|X}(c \mid \boldsymbol{x}) = \frac{p_Y(c) \cdot p'_{X|Y}(\boldsymbol{x} \mid c)}{\sum_{c=1}^{C} p_Y(c) \cdot p'_{X|Y}(\boldsymbol{x} \mid c)} = \frac{p'_{(X,Y)}(\boldsymbol{x}, c)}{p_X(\boldsymbol{x})}. \tag{A.7}$$

Therefore, the underlying distribution for synthetic datasets generated through the introduction of noise using the proximity approach remains the same as for the datasets using the MVN distribution with updated means.

A.1.3. Ground truth MI

By plugging in the conditional $p_{Y|X}(\cdot)$ and the marginal $p_Y(\cdot)$ defined above in (2), the ground truth MI for the generated system dataset $D = \{\boldsymbol{x}_i, y_i\}$ is approximated as

$$GI(D) \approxeq \frac{1}{N}\sum_{i=1}^{N} p_{Y|X}(y_i \mid \boldsymbol{x}_i)\log\left(p_{Y|X}(y_i \mid \boldsymbol{x}_i)\right) - \sum_{c=1}^{C} p_Y(c)\log\left(p_Y(c)\right)\,. \tag{A.8}$$

For systems simulated using perturbation and proximity techniques, the modified conditional $p_{Y_\epsilon|X}(\cdot)$ in (A.6) and $p'_{Y|X}(c \mid \boldsymbol{x})$ in (A.7) and the unchanged marginal on Y is used to calculate the ground truth MI.

A.2. Implementation details

This section details the implementation setup, including automated tools and calibration methods used for accurate MI estimation via LOG-LOSS. Hyperparameters and their ranges for TabPFN, AutoGluon, baseline MI estimators, and ILD approaches are provided in Table A.5, used for experiments in Sections 3 and 4.2. Python code including MI estimation, ILD methods, synthetic dataset generation, and the OpenML parser, is available in the GitHub repository[5], along with the scripts used for running the experiments in another GitHub repository[6]. The 10 IL-Datasets per time delay were transformed into multi-class datasets with 11 padding classes and uploaded to OpenML.[4]

A.2.1. Performance evaluation metrics

Koyejo et al. [44] defines evaluation measures used for evaluating the performance of classifiers, using the ground truth labels denoted by $\boldsymbol{y} = (y_1, \ldots, y_N)$ for a given $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and the predictions denoted by the vector $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_N)$, acquired from the *empirical risk minimizers* g and g_p (cf. Section 2.2.2).

Accuracy and error-rate The accuracy is defined as the proportion of correct predictions, while error-rate is the proportion of incorrect predictions

$$\mathrm{m}_{\mathrm{ACC}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) := \frac{1}{N}\sum_{i=1}^{N}[\![y_i = \hat{y}_i]\!]\,, \quad \mathrm{m}_{\mathrm{ERR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) := \frac{1}{N}\sum_{i=1}^{N}[\![y_i \neq \hat{y}_i]\!]\,.$$

Confusion matrix (CM) The CM is defined using the *true positive* (m_{TP}), *true negative* (m_{TN}), *false positive* (m_{FP}), and *false negative* (m_{FN}) as

$$\mathrm{m}_{\mathrm{CM}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \begin{bmatrix} \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N}[\![y_i = 0, \hat{y}_i = 0]\!] & \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N}[\![y_i = 0, \hat{y}_i = 1]\!] \\ \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N}[\![y_i = 1, \hat{y}_i = 0]\!] & \mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N}[\![y_i = 1, \hat{y}_i = 1]\!] \end{bmatrix}$$

False negative rate (FNR) and FPR The FNR (type 2 error) is defined as the ratio of *false negatives* to the total number of positive instances, while the FPR (type 1 error) is the ratio of *false positives* to the number of negative instances, defined as

$$\mathrm{m}_{\mathrm{FNR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}\,, \quad \mathrm{m}_{\mathrm{FPR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}\,.$$

A.2.2. AutoML tools

In the last decade, the field of AutoML has emerged to address the growing need for automating machine learning pipelines, particularly in the absence of expert intervention. While AutoML aims to streamline the entire data science workflow, much of the research has focused on the challenge of the combined algorithm selection and hyperparameter optimization (CASH) problem. Since then, various systems have been devised, demonstrating promising performances for tailoring the choice of ML algorithms and the setting of their hyperparameters to a given task, typically comprising a dataset and a loss function [45]. While various AutoML systems with complementary strengths have been proposed in the literature, a recent benchmark study [46] suggests AutoGluon [39] as the AutoML system with the best performance across datasets and different tasks. In Hollmann et al. [47], instead of tackling the CASH problem, a general predictor called TabPFN is fitted across various datasets and can immediately return highly accurate predictions. Yielding competitive performance to AutoGluon, TabPFN suggests itself as a state-of-the-art AutoML tool that returns results substantially faster than other AutoML systems. Therefore, we selected AutoGluon and TabPFN to approximate the Bayes predictor in our methods for estimating MI and detecting ILs in systems [39,47].

AutoGluon This AutoML tool allows the customization of the set of considered ML methods, including gradient boosting machine (GBM) tree-based models and deep neural networks (NNs). For our study, we limited the search space of AutoGluon to consistent classifiers, i.e., tree-based ensemble classifiers like RF, XT, GBM algorithms (LightGBM, CatBoost, XGBoost), and MLP implemented using `PyTorch` and `fastai` [48,49]. The AutoGluon search space comprises various learning algorithms with their corresponding hyperparameter ranges, is listed in Table A.5.

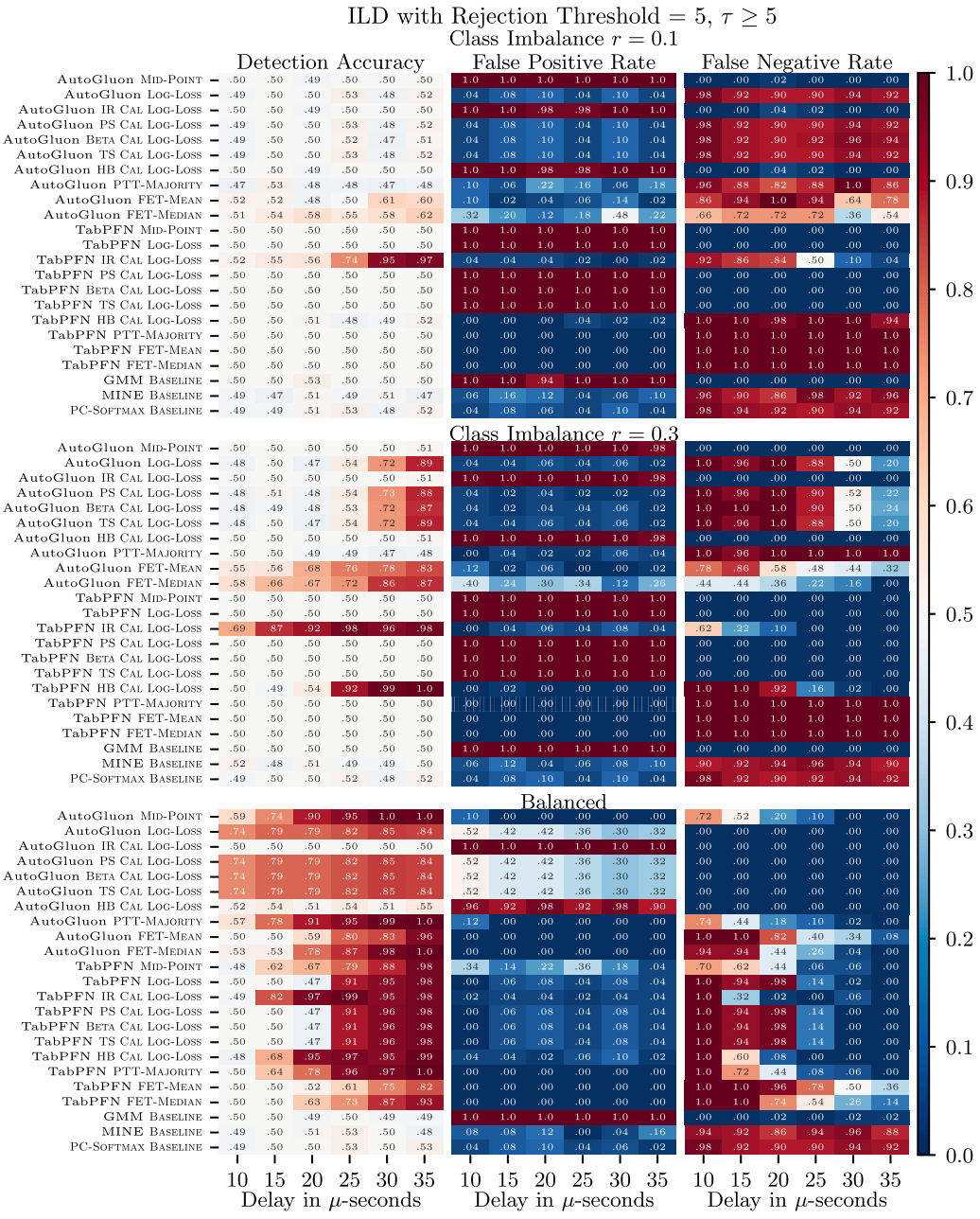


Fig. 10. Performance of ILD approaches versus time delay with 5 µs step.

TabPFN This AutoML tool supports up to 100 numeric features, so we apply dimensionality reduction, using RF or XT when $d > 100$. We fine-tune TabPFN using HPO, optimizing parameters such as the number of reduced features, the reduction method, and the number of prior-fitted models, with hyperparameter ranges listed in Table A.5.

Appendix B. Generalization capability of ILD approaches

This section evaluates the generalization performance of our ILD approaches against state-of-the-art baselines. We assess ILD methods with respect to time delay—the computation time difference between correct and incorrect messages—capturing the system's LAS. We focus on detection accuracy, FPR, and FNR for identifying timing side-channels targeting Bleichenbacher-style attacks. Heatmaps with a fixed rejection threshold of 5 ($\tau \geq 5$) are used to visualize results: Fig. 10 uses linear 5 µs increments, while Fig. 11 uses logarithmic 2 µs steps.

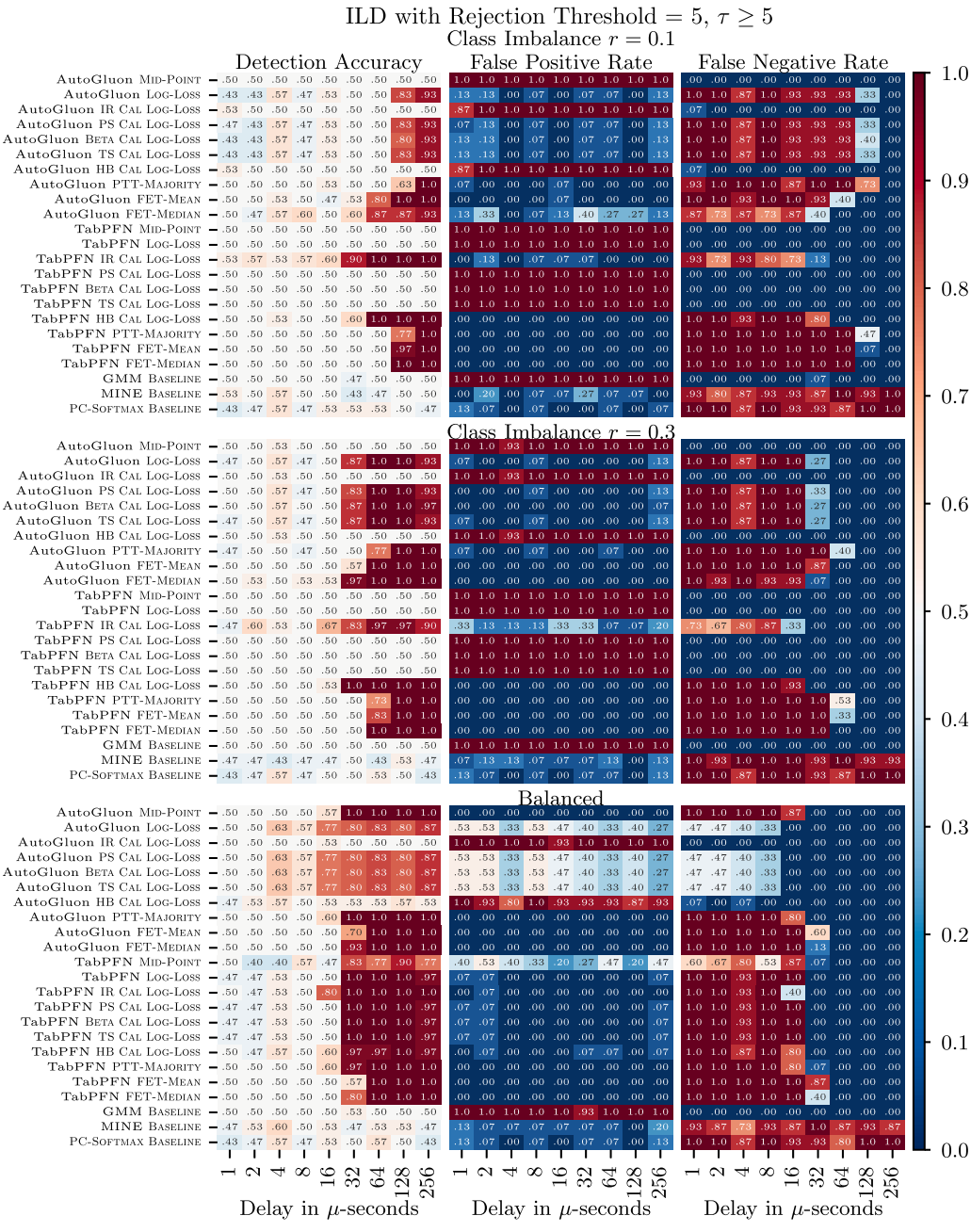


Fig. 11. Performance of ILD approaches versus time delay with 2 µs logarithmic step.

B.1. TabPFN

This section analyzes MI and classification-based ILD approaches using TabPFN. In summary, TabPFN IR Cal Log-Loss generalizes well, detecting 60 % of ILs at minimal delays of 20 µs. It performs even better on balanced datasets, achieving up to 80 % accuracy.

Classification-based approaches Detection accuracy drops sharply for all approaches when time delays fall below a critical threshold (35 µs for imbalanced dataset and 16 µs for balanced dataset), with all methods failing to detect ILs and yielding a 100 % FNR. FET-Median performs best on imbalanced datasets, while PTT-Majority leads on balanced ones.

MI-based approaches Overall, IR Cal Log-Loss consistently outperforms other approaches across both balanced and imbalanced datasets, with HB Cal Log-Loss showing competitive results. While most approaches overfit and overestimate MI on imbalanced data, IR Cal Log-Loss and HB Cal Log-Loss maintain reliable detection and avoid false positives. Both methods generalize well,

detecting more than 80 % of ILs, even under minimal time delays of 20 µs, specifically for the balanced datasets, underscoring IR CAL LOG-LOSS's strength in enhancing LOG-LOSS.

B.2. AutoGluon

This section analyzes MI and classification-based ILD approaches using AutoGluon. Generally, they detect non-existent ILs in systems generating imbalanced datasets, especially with time delays under 20 µs. However, in systems generating balanced datasets, they occasionally detect over 70 % of ILs with similar short delays.

Classification-based approaches Overall, FET based approaches show almost comparable detection performance in systems generating balanced and imbalanced datasets, while PTT-MAJORITY performs better in the case of balanced systems datasets. All approaches reliably detect ILs for time-delays above a critical threshold, with all methods failing to detect ILs and yielding a 100 % FNR, below this threshold (30 µs for imbalanced dataset and 16 µs for balanced dataset)).

MI-based approaches All methods reliably detect ILs above a critical time-delay threshold, but fail below it—yielding a 100 % FNR (at 25 µs for imbalanced and 10 µs for balanced datasets). The midpoint approach performs best on balanced datasets, while LOG-LOSS excels on imbalanced ones. This suggests that calibration (CAL LOG-LOSS) does not improve log-loss-based MI estimation; in fact, IR CAL LOG-LOSS and HB CAL LOG-LOSS degrade it by overfitting and overestimating MI, resulting in false positives and a 100 % FPR—consistent with our findings in Sections 3.3.1 and 4.3.

B.3. Baselines

The baselines consistently achieve around 50 % accuracy on both balanced and imbalanced datasets, underscoring the need for specialized ILD approaches. MINE and PC-SOFTMAX frequently miss ILs, with detection occasionally reaching 60 %. GMM tends to overestimate MI, resulting in false positives and a 100 % FPR due to overfitting in high-dimensional settings.

B.4. Summary

Overall, we conclude that the calibration techniques (CAL LOG-LOSS) have mixed effects on LOG-LOSS performance across TabPFN and AutoGluon. While IR CAL LOG-LOSS and HB CAL LOG-LOSS often degrade AutoGluon LOG-LOSS performance due to overfitting, they improve detection with TabPFN—especially IR CAL LOG-LOSS, which consistently achieves the best accuracy on both balanced and imbalanced datasets. In contrast, with AutoGluon, FET-MEDIAN performs best on imbalanced datasets, while MID-POINT and PTT-MAJORITY are most effective for balanced datasets. These findings, consistent with results in Sections 3.3 and 4.2, highlight the importance of selecting calibration methods based on the dataset and underlying model. The MID-POINT approach excels on balanced datasets but falsely detects ILs in imbalanced ones, resulting in a 100 % FPR, confirming the intuition in Section 3.1.1. Baseline ILD approaches remain suboptimal detecting only 50 % of ILs, often with 100 % FPR or FNR.

Data availability

The data is uploaded on OpenML: https://www.openml.org/s/383.

References

[1] J. Kelsey, Compression and information leakage of plaintext, in: Fast Software Encryption, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 263–276.
[2] B. Hettwer, S. Gehrer, T. Güneysu, Applications of machine learning techniques in side-channel attacks: a survey, J. Cryptogr. Eng. 10 (2019) 135–162, https://doi.org/10.1007/s13389-019-00212-8.
[3] A. Shabtai, Y. Elovici, L. Rokach, A Survey of Data Leakage Detection and Prevention Solutions, 1 ed., Springer US, New York, NY, 2012.
[4] K. Chatzikokolakis, T. Chothia, A. Guha, Statistical measurement of information leakage, in: Tools and Algorithms for the Construction and Analysis of Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 390–404.
[5] S. Gao, G. Ver Steeg, A. Galstyan, Efficient estimation of mutual information for strongly dependent variables, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 38, PMLR, San Diego, California, USA, 2015, pp. 277–286.
[6] K.R. Moon, K. Sricharan, A.O. Hero, Ensemble estimation of generalized mutual information with applications to genomics, IEEE Trans. Inf. Theory 67 (2021) 5963–5996, https://doi.org/10.1109/TIT.2021.3100108.
[7] S. Picek, G. Perin, L. Mariot, L. Wu, L. Batina, Sok: deep learning-based physical side-channel analysis, ACM Comput. Surv. 55 (2023), https://doi.org/10.1145/3569577.
[8] T. Moos, F. Wegener, A. Moradi, DL-LA: deep learning leakage assessment, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2021 (2021) 552–598, https://doi.org/10.46586/tches.v2021.i3.552-598.
[9] T. Perianin, S. Carré, V. Dyseryn, A. Facon, S. Guilley, End-to-end automated cache-timing attack driven by machine learning, J. Cryptogr. Eng. 11 (2020) 135–146, https://doi.org/10.1007/s13389-020-00228-5.
[10] V. Cristiani, M. Lecomte, P. Maurine, Leakage assessment through neural estimation of the mutual information, in: Lecture Notes in Computer Science, Springer International Publishing, Berlin, Heidelberg, 2020, pp. 144–162.
[11] Z. Qin, D. Kim, Rethinking softmax with cross-entropy: neural network classifier as mutual information estimator, CoRR, arXiv:1911.10688, 2019.
[12] J. Zhang, M. Zheng, J. Nan, H. Hu, N. Yu, A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020 (2020) 73–96, https://doi.org/10.46586/tches.v2020.i3.73-96.

[13] S. Picek, A. Heuser, A. Jovic, S. Bhasin, F. Regazzoni, The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019 (2018) 209–237, https://doi.org/10.13154/tches.v2019.i1.209-237.
[14] P. Gupta, A. Ramaswamy, J. Drees, E. Hüllermeier, C. Priesterjahn, T. Jager, Automated information leakage detection: a new method combining machine learning and hypothesis testing with an application to side-channel detection in cryptographic protocols, in: Proceedings of the 14th International Conference on Agents and Artificial Intelligence, INSTICC, SCITEPRESS - Science and Technology Publications, Virtual Event, 2022, pp. 152–163.
[15] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, 2005, pp. 13–55.
[16] V. Vapnik, Principles of risk minimization for learning theory, in: Proceedings of the 4th International Conference on Neural Information Processing Systems, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 831–838.
[17] C.M. Bishop, Probability Distributions, Springer, New York, NY, 2006.
[18] T. Gneiting, A.E. Raftery, Strictly proper scoring rules, prediction, and estimation, J. Am. Stat. Assoc. 102 (2007) 359–378, https://doi.org/10.1198/016214506000001437.
[19] L. Devroye, L. Györfi, G. Lugosi, The Bayes Error, vol. 31, Springer, New York, NY, 1996, pp. 9–20.
[20] M.I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, D. Hjelm, Mutual information neural estimation, in: Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 531–540.
[21] F. Maia Polo, R. Vicente, Effective sample size, dimensionality, and generalization in covariate shift adaptation, Neural Comput. Appl. 35 (2022) 18187–18199, https://doi.org/10.1007/s00521-021-06615-1.
[22] D. Tebbe, S. Dwyer, Uncertainty and the probability of error (corresp.), IEEE Trans. Inf. Theory 14 (1968) 516–518, https://doi.org/10.1109/tit.1968.1054135.
[23] M. Hellman, J. Raviv, Probability of error, equivocation, and the Chernoff bound, IEEE Trans. Inf. Theory 16 (1970) 368–372, https://doi.org/10.1109/tit.1970.1054466.
[24] R.M. Fano, Transmission of Information: A Statistical Theory of Communications, The MIT Press, Cambridge, MA, 1961.
[25] M.-J. Zhao, N. Edakunni, A. Pocock, G. Brown, Beyond Fano's inequality: bounds on the optimal f-score, ber, and cost-sensitive risk and their implications, J. Mach. Learn. Res. 14 (2013) 1033–1090.
[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, June 27-30, IEEE Computer Society, 2016, pp. 2818–2826.
[27] T. Silva Filho, H. Song, M. Perello-Nieto, R. Santos-Rodriguez, M. Kull, P. Flach, Classifier calibration: a survey on how to assess and improve predicted class probabilities, Mach. Learn. 112 (2023) 3211–3260, https://doi.org/10.1007/s10994-023-06336-7.
[28] H. Kario, Everlasting ROBOT: the marvin attack, in: G. Tsudik, M. Conti, K. Liang, G. Smaragdakis (Eds.), Computer Security – ESORICS 2023: 28th European Symposium on Research in Computer Security, The Hague, the Netherlands, September 25–29, 2023, Proceedings, Part III, in: Lecture Notes in Computer Science, vol. 14137, Springer Nature Switzerland, The Hague, the Netherlands, 2024, pp. 243–262.
[29] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (1979) 65–70.
[30] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[31] C. Nadeau, Inference for the generalization error, Mach. Learn. 52 (2003) 239–281, https://doi.org/10.1023/a:1024068626366.
[32] R.A. Fisher, On the interpretation of χ 2 from contingency tables, and the calculation of p, J. R. Stat. Soc. 85 (1922) 87, https://doi.org/10.2307/2340521.
[33] D. Chicco, N. Tötsch, G. Jurman, The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation, BioData Min. 14 (2021) 13, https://doi.org/10.1186/s13040-021-00244-z.
[34] B. Bhattacharya, D. Habtzghi, Median of the p value under the alternative hypothesis, Am. Stat. 56 (2002) 202–206, https://doi.org/10.1198/000313002146.
[35] D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs #1, in: Advances in Cryptology — CRYPTO '98, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 1–12.
[36] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, E. Tews, Revisiting SSL/TLS implementations: new bleichenbacher side channels and attacks, in: 23rd USENIX Security Symposium (USENIX Security 14), USENIX Association, San Diego, CA, 2014, pp. 733–748.
[37] D. Funke, Pushing the AutoSCA tool to picosecond precision: improving timing side channel detection, https://doi.org/10.13140/RG.2.2.33070.08005, 2022.
[38] D. Salinas, N. Erickson, Tabrepo: a large scale repository of tabular model evaluations and its automl applications, in: K. Eggensperger, R. Garnett, J. Vanschoren, M. Lindauer, J.R. Gardner (Eds.), Proceedings of the Third International Conference on Automated Machine Learning, in: Proceedings of Machine Learning Research, vol. 256, PMLR, 2024, pp. 19/1–30, https://proceedings.mlr.press/v256/salinas24a.html.
[39] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, A. Smola, AutoGluon-tabular: robust and accurate AutoML for structured data, arXiv preprint, arXiv:2003.06505, 2020.
[40] D.R. Brillinger, Some data analyses using mutual information, Braz. J. Probab. Stat. 18 (2004) 163–182, http://www.jstor.org/stable/43601047.
[41] F. Armknecht, C. Boyd, G.T. Davies, K. Gjøsteen, M. Toorani, Side channels in deduplication: trade-offs between leakage and efficiency, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17), Association for Computing Machinery, New York, NY, USA, 2017, pp. 266–274.
[42] Y. Zhou, D. Feng, Side-channel attacks: ten years after its publication and the impacts on cryptographic module security testing, IACR Cryptology ePrint Archive, http://eprint.iacr.org/2005/388, 2005.
[43] S. Faezi, R. Yasaei, A. Barua, M.A.A. Faruque, Brain-inspired golden chip free hardware trojan detection, IEEE Trans. Inf. Forensics Secur. 16 (2021) 2697–2708, https://doi.org/10.1109/TIFS.2021.3062989.
[44] O. Koyejo, P. Ravikumar, N. Natarajan, I.S. Dhillon, Consistent multilabel classification, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, MIT Press, Cambridge, MA, USA, 2015, pp. 3321–3329.
[45] M. Zöller, M.F. Huber, Benchmark and survey of automated machine learning frameworks, J. Artif. Intell. Res. 70 (2021) 409–472, https://doi.org/10.1613/jair.1.11854.
[46] P. Gijsbers, M.L.P. Bueno, S. Coors, E. LeDell, S. Poirier, J. Thomas, B. Bischl, J. Vanschoren, Amlb: an automl benchmark, J. Mach. Learn. Res. 25 (2024) 1–65.
[47] N. Hollmann, S. Müller, K. Eggensperger, F. Hutter, TabPFN: a transformer that solves small tabular classification problems in a second, in: The Eleventh International Conference on Learning Representations, 2023, https://openreview.net/forum?id=cp5PvcI6w8_.
[48] G. Biau, L. Devroye, G. Lugosi, Consistency of random forests and other averaging classifiers, J. Mach. Learn. Res. 9 (2008) 2015–2033.
[49] J. Mielniczuk, J. Tyrcha, Consistency of multilayer perceptron regression estimators, Neural Netw. 6 (1993) 1019–1022, https://doi.org/10.1016/s0893-6080(09)80011-7.