Advanced Machine Learning Methods for Information Leakage Detection in Cryptographic Systems

Pritha Gupta

May 20, 2025

PADERBORN UNIVERSITY
The University for the Information Society
Department of Electrical Engineering,
Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

Software Innovation Campus Paderborn
Research Institute
Paderborn University
Zukunftsmeile 2
33102 Paderborn

Dissertation

In partial fulfillment of the requirements for the academic degree of
Doctor rerum naturalium (Dr. rer. nat.)

Advanced Machine Learning Methods for Information Leakage Detection in Cryptographic Systems

Pritha Gupta

1. Reviewer	Prof. Dr. Eyke Hüllermeier Künstliche Intelligenz und Maschinelles Lernen Ludwig-Maximilians-Universität, München
2. Reviewer	Prof. Dr. Juraj Somorovsky Department of Computer Science Paderborn University
Supervisor	Prof. Dr. Eyke Hüllermeier

May 20, 2025

Pritha Gupta

Advanced Machine Learning Methods for Information Leakage Detection in Cryptographic Systems

Supervisor: Prof. Dr. Eyke Hüllermeier
Reviewers: Prof. Dr. Eyke Hüllermeier and Prof. Dr. Juraj Somorovsky

PhD Mentor: Dr. Marcel Wever
Research Mentor: Dr. Karlson Pfannschmidt
Postdoc Advisor: Prof. Dr. Ivan Habernal

Paderborn University
Software Innovation Campus Paderborn (SICP)

Department of Electrical Engineering
Computer Science and Mathematics
Zukunftsmeile 2
33102 Paderborn

Abstract

In today's data-driven world, the proliferation of public information exacerbates the challenge of information leakage (IL), risking the exposure of sensitive data through observable system outputs. Traditional statistical methods, reliant on mutual information (MI) estimation, often face significant computational complexities and the curse of dimensionality. Meanwhile, emerging supervised learning approaches lack a theoretical foundation and are limited to domain-specific applications with balanced binary data.

In this work, I developed a theoretical framework integrating statistical learning and information theory to effectively quantify and detect IL in cryptographic systems. The proposed generalized measure leakage assessment score (LAS) quantifies IL by leveraging Bayes predictor's performance to estimate MI through Log-Loss and the rank of the key byte of the AES-encrypted systems. Leveraging the consistency of automated machine learning (AutoML), benchmark tools are employed to approximate Bayes predictor's performance, providing robust LAS estimates, even for systems with extensive key spaces. Proposed approaches employ statistical tests on MI estimates, confusion matrices (CMs), and accuracies obtained from AutoML tools to detect IL. These tests are performed multiple times, and their results are aggregated using the Holm-Bonferroni correction to ensure reliable and confident decisions regarding the presence of IL.

Experimental results on synthetic and real-world OpenSSL TLS servers, vulnerable to Bleichenbacher's side-channel attack (SCA), show that the proposed approaches outperform baseline methods. For AES-encrypted systems, IL is

quantified using metrics, also providing insights on the susceptibility to template SCAs. The proposed automated black-box approaches use benchmark AutoML tools to optimize convolutional neural networks (CNNs) architectures, identifying RANDOM search as the most effective for identifying system vulnerabilities. However, the performance variability of these tools necessitates further improvements for comprehensive security analysis of cryptographic systems.

Kurzfassung

In der heutigen datengesteuerten IT-Landschaft verschärft die Verbreitung öffentlicher Informationen die Herausforderung von Datenlecks und birgt die Gefahr, dass sensible Daten unbeabsichtigt durch beobachtbare Systemausgaben offengelegt werden. Herkömmliche statistische Methoden zur Detektion von Informationslecks, die auf der Schätzung von gegenseitiger Information beruhen, sind oft mit erheblichem Rechenaufwand und dem "Fluch der Dimensionalität" konfrontiert. Auf der anderen Seite mangelt es den neuen Ansätzen des überwachten maschinellen Lernens an einer theoretischen Grundlage und sie sind auf domänenspezifische Anwendungen mit ausgeglichenen binären Daten beschränkt.

In dieser Arbeit habe ich einen theoretischen Rahmen entwickelt, der statistisches Lernen und Informationstheorie integriert, um Informationslecks in kryptographischen Systemen zu quantifizieren und effektiv zu erkennen. Die vorgeschlagene verallgemeinerte Metrik LAS quantifiziert Informationslecks, indem es die Leistung des Bayes-Klassifikators nutzt, um die gegenseitige Information durch Kreuz-Entropie-Verlust (Log-Loss) und den Rang des Schlüsselbytes der AES-verschlüsselten Systeme zu schätzen. Mithilfe der Konsistenz von AutoML werden Benchmark-Tools zur Approximation der Leistung des Bayes-Klassifikators eingesetzt.Dies liefert robuste LAS-Schätzungen, die auch Systeme mit umfangreichen Schlüsselräumen effektiv adressieren. Die vorgeschlagenen Ansätze verwenden statistische Tests auf Schätzungen der gegenseitigen Information, Konfusionsmatrizen und Treffergenauigkeiten (Accuracy), die von AutoML-Tools erhalten wurden, um Informationslecks zu erkennen. Diese Tests werden mehrfach durchgeführt, und ihre Ergebnisse

werden mithilfe der Holm-Bonferroni-Korrektur aggregiert, um zuverlässige und sichere Entscheidungen über das Vorhandensein von Informationslecks zu gewährleisten.

Experimentelle Ergebnisse auf synthetischen und realen Datensätzen, erstellt mit für Bleichenbacher's Seitenkanalangriff verwundbaren OpenSSL TLS-Servern, zeigen, dass die vorgeschlagenen Ansätze die bisherigen Methoden übertreffen. Bei AES-verschlüsselten Systemen werden Informationslecks durch Hardware-Seitenkanäle mithilfe von Metriken quantifiziert, welche auch die Anfälligkeit für Template-Angriffe auswerten. Die vorgeschlagenen automatisierten Blackbox-Ansätze verwenden Benchmark-AutoML-Tools zur Optimierung verschiedener CNN-Architekturen und zeigen, dass die zufällige Suchstrategie am effektivsten zur Identifizierung von Systemschwächen ist. Die Leistungsschwankungen dieser Tools machen jedoch weitere Verbesserungen für eine umfassende Sicherheitsanalyse kryptographischer Systeme erforderlich.

Dedication

This dissertation is dedicated, in memoriam, to my mentor, Sumedha Uniyal, who passed away in February 2020 and was the source of my inspiration.

Your wisdom, integrity, and unwavering dedication to pursuing knowledge have shaped my academic journey and the person I have become. You taught me the foundations of mathematics and computer science, instilling the curiosity and resilience that have guided me through every challenge. Your legacy lives on through your work and the countless lives you touched, including mine.

Every equation solved, every algorithm understood, and every breakthrough achieved are not merely milestones in my career but reflections of your influence and guidance. You taught me that the pursuit of knowledge is endless, and I will forever carry the lessons of perseverance, brilliance, and humility that you embodied.

This work is for you, with my deepest gratitude and admiration.

Acknowledgments

I want to express my heartfelt gratitude to my supervisor, Prof. Dr. Eyke Hüllermieier, for his invaluable guidance throughout this journey. Eyke provided me with countless opportunities, from fruitful collaborations to giving me the freedom and space to develop my research. His attention to detail, intuitive approach to concepts, and ability to explain complex ideas with clarity have set a standard I hope to match someday.

I want to thank Eyke again and Prof. Dr. Juraj Somorovsky for their invaluable feedback on the thesis structure, detailed review, meticulous proofreading, and insightful guidance throughout the review process. Thank you, Juraj, for always motivating me, answering my endless security-related questions, and engaging in deep discussions about the field. Your interest in my work during SICP events, such as Tag der IT-Sicherheit, is greatly appreciated.

I am deeply grateful to my PhD mentor, Dr. Marcel Wever, for his invaluable guidance and support, which provided the perfect framework for my research and motivated me during challenging times. I am deeply grateful to Dr. Karlson Pfandschmidt, my research mentor, for his patience and support as a research student in the ISML group at Paderborn University under Eyke. I extend my heartfelt gratitude to my collaborator, Dr. Jan Peter Drees, for his invaluable insights into security issues solvable through machine learning methods and his dedicated support in proofreading this thesis. Special thanks to Dennis Funke, who, under Jan's supervision, simulated real-world OpenSSL TLS servers for his bachelor's thesis, providing a practical problem to solve. I am also grateful to my thesis students—Sebastian Silva, Priyanka Roy, Varun Nand Kumar Golani, Natalie Weiß, Jiawen Wang, and Louis Wang—whose dedication enriched my

research and helped me grow as an advisor. I am grateful to Stefan Heid, Karim Belaid, and Marcel for proofreading my dissertation and to Dr. Björn Haddenhorst for repeatedly reviewing the formal sections of my papers and thesis.

Special thanks to the SICP team, especially Stefan Sauer and Gunnar Schomaker, for providing a spacious office at ZM2 after Eyke's move to Munich. I appreciate your open-door policy, procedural guidance, and efforts to foster unity and collaboration. The open workspace enabled me to focus on my dissertation and maintain balance through in-office sports with my mentor's "SPARTANS". Additionally, I want to thank Sonja Saage, Gabriele Stall, Elisabeth Lengeling, Christina Lange, and Markus Franke for their invaluable organizational support during my time at the SICP, Paderborn University. Cheers to the "ZM2 gang"—for lunch breaks filled with juggling, cube-solving, and endless laughs! Special thanks go to the itemis AG team members, Nils Weidmann, Christoph Borowski, and Dennis Röck, for always cheering me up when I was "GRUMPY" and providing the best coffee during my writing phase. Thanks to Jasmin Brandt, Stefan Heid, and Karlson, the ZM2 office was always lively and fun, primarily because of our memorable runs and bouldering sessions after work.

This dissertation was funded by BMBF (16KIS1190, AutoSCA[1]) and ERC-802823. Computing time on Noctua2 at the NHR Center PC[2], Paderborn University, is also acknowledged.

I am incredibly grateful to my extended family for caring for everything in India on my behalf. I owe a great deal of gratitude to my parents for making me who I am today and to my sister, Shambhavi Gupta, for her unconditional love and encouragement, specifically during the writing phase of my dissertation. Most importantly, I want to express my most profound appreciation to my mentor, Prajna Uniyal, for his tremendous support, unwavering belief in me, and productive firmness that pushed me forward whenever I was down.

[1] `https://www.sicp.de/projekte/abgeschlossene-projekte/autosca`

Contents

1. Introduction

The problem of information leakage (IL) has become a significant and complex challenge in today's data-driven world [178]. The rapid proliferation of publicly available data, coupled with the increasing use of internet of things (IoT) technologies, has magnified the threat of IL, posing a substantial risk to the security and confidentiality of systems [178, 105]. This heightened threat of IL is amplified by the increasing use of IoT technologies, posing substantial risks to the confidentiality of cryptographic systems.

IL occurs when sensitive or confidential information is inadvertently exposed to unauthorized individuals through observable information of a system [86]. This unintended disclosure of sensitive data can lead to severe consequences, ranging from potential electrical blackouts to the theft of critical information such as medical records and military secrets [86, 178]. Consequently, the efficient detection and quantification of IL are paramount. The theoretical foundations of quantifying IL in systems stem from the field of information theory and propose to estimate the mutual information (MI) between observable and secret information [28]. MI is shown to be challenging to compute specifically for high-dimensional data, despite being a pivotal measure [69, 51]. Traditional solutions often resort to using statistical estimation to calculate MI, but usually encounter challenges when dealing with high-dimensional data, a problem known as the *curse of dimensionality* [69, 111]. Recent works offer more robust non-parametric approaches using Kernel density estimation (KDE) or K-nearest neighbor (KNN) with improved convergence rates for MI estimation in complex

datasets [182, 101, 135]. However, their implementation in high-dimensional scenarios remains challenging due to the sophisticated statistical techniques involved [101].

In recent years, machine learning (ML) techniques have become very popular in the field of information leakage detection (ILD), particularly in the sub-field of performing side-channel attacks (SCAs) on cryptographic systems [150]. These systems unintentionally release the *observable information* via many modes, such as network messages, CPU caches, power consumption, or electromagnetic radiation called the *side channels* [150]. These modes are exploited by the SCAs to reveal the secret inputs (secret keys, plaintexts) to the adversaries, potentially rendering all implemented cryptographic protections irrelevant [136, 86]. A system is considered to contain a side channel if it is susceptible to SCAs, thereby making it *vulnerable*. Therefore, detecting the existence of a side channel in the communication channel of a cryptographic system is equivalent to uncovering IL [86].

In this field, the most relevant literature uses ML to perform SCAs, not preventing side channels through early detection of ILs in AES-encrypted systems [86]. Notably, current ML-based methods in this realm detect side channels, preventing SCAs and protecting the system on both algorithmic and hardware levels [137, 136]. These approaches leverage the observable information of a system to classify it as vulnerable (with IL) or non-vulnerable (without IL) [142, 137]. These techniques extract observable information from secure systems used as input, categorizing the systems as non-vulnerable (labeled as 0). Subsequently, they intentionally introduce known ILs in these systems, categorizing them as vulnerable (labeled as 1). Finally, the extracted observable information with the corresponding class label is used to produce the binary classification dataset for the learning model. However, this approach confines these techniques to domain-specific scenarios and cannot be easily transferred to detect other unknown leakage patterns [142].

ML-based techniques have shown promising potential in estimating MI within classification datasets generated by systems [12, 158, 39]. Compared to traditional estimators using KDE or KNN to approximate MI, mutual information neural estimation (MINE) is more scalable for high-dimensional settings but may require more careful tuning [12, 40]. Despite this, MINE grapple with challenges related to convergence issues and high computational complexity [12, 39, 40]. For instance, the approach proposed by [158] introduces a lower bound on MI using the Kullback-Leibler (KL) divergence. However, this method underestimates MI and might not capture specific subclasses of IL. Recent advancements have demonstrated the effectiveness of ML-based techniques in directly detecting ILs by analyzing the accuracy of the supervised learning models on extracted system data [136, 49]. Yet, these methods exhibit limitations in handling imbalanced and noisy datasets, commonly encountered in practical scenarios, and tend to miss ILs by producing false negatives [207, 148].

Since most of the current approaches employing deep learning (DL) for performing the SCA on AES-encrypted systems, to infer the occurrence of IL suffer from convergence issues and finding an appropriate architecture for the network [136, 40]. To address these issues, neural architecture search (NAS) (subset of automated machine learning (AutoML)) approach, which aims to find an optimal neural network architecture, has gained attention in performing the automated SCA on AES-encrypted systems [201, 169, 1]. Though effective, these approaches utilize the attack dataset, including the information about the secret key for finding the architecture, which risks overfitting and data-snooping, leading to inflated performance estimates that may not generalize to real-world, unlabeled datasets available in black-box settings. InfoNEAT [1] evolves architectures and hyperparameters simultaneously using one-versus-rest (OVR) classification suited for template SCA, though it is not directly comparable to single-architecture methods due to its distinct output structure. Also, usage of OVR classification makes it computationally time-inefficient to perform ILDs or SCAs. Recently, automated solutions for side-channel analysis or ILD have been proposed, focusing on specific aspects such as feature selection, MI estimation for univariate side or observable information, or leakage simulation [194,

159, 9]. However, these remain limited in scope, underscoring the need for a comprehensive, end-to-end automated solution for performing ILD applicable to any multivariate observable information.

The goal of this thesis is to establish a comprehensive framework for automated ILD formalized using a connection between MI and Bayes predictor to define a generalized measure called leakage assessment score (LAS) which is consequently utilized to detect and quantify IL in systems producing classification datasets. I also define a metric called vulnerability score (VS) based on the LAS using guessing entropy (GE) (specific for secret key byte as sensitive information) in conjunction with trace sufficiency threshold (TST), which is used to assess IL in AES-encrypted systems, improving precision in assessing system susceptibility to Template SCAs. I propose to use the automated machine learning (AutoML) (super-set of NAS) approaches to induce Bayes predictor and propose two new MI estimation methods, based on Log-Loss and accuracy of AutoML-induced Bayes predictor.

To detect IL in systems with sensitive information represented as class labels (e.g., classification tasks with fewer classes), I propose using statistical test-based approaches, utilizing one-sample t-test (OTT) on MI, Fisher's exact test (FET) on confusion matrices (CMs), and paired t-test (PTT) on accuracy estimates of AutoML-induced Bayes predictor. This framework is validated through extensive empirical studies, benchmarking the proposed ILD approaches against state-of-the-art methods for detecting ILs via the processing time and error code side information in network trace side channels, making the system particularly vulnerable towards the Bleichenbacher's SCAs.

However, in practical scenarios such as AES-encrypted systems, where template SCAs target a single byte (256 classes) of the 16-byte or 32-byte secret key, these statistical approaches often face limitations due to the curse of dimensionality of noisy power consumption traces, leading to imprecise estimated scores for a large number of classes. To address these challenges, I propose a black-box NAS approach is proposed, benchmarking NAS-generated convolutional

neural networks (CNNs) against fixed architectures for superior adaptability and efficiency in detecting hardware-based vulnerabilities or ILs, countering template SCAs on AES-encrypted systems.

The remainder of this chapter is organized as follows: Section 1.1 provides a detailed overview of the related work, motivating the research questions answered via the contributions to this thesis. Section 1.2 discusses this research's potential commercial integration and impact, outlines the thesis, and presents co-author contribution statements for the publications on which this work is partly based. Finally, Section 1.3 introduces the notations and standardized visual formats used throughout the diagrams in this thesis. In line with standard academic practice, I will use the first person singular ("I") throughout this thesis to reflect my role as an independent researcher while acknowledging the valuable contributions of Jan Peter Drees, Marcel Wever, Arunselvan Ramaswamy, and Eyke Hüllermeier.

1.1. Motivation

This work is motivated by the detailed efforts of security experts leveraging ML methods, specifically the DL ones, to ensure system privacy, often without fully grasping the theoretical underpinnings of why these approaches perform well in performing ILD in cryptographic systems. To do so, I present a detailed overview of the related work for performing ILD in cryptographic systems, specifically those that communicate via OpenSSL TLS servers and are encrypted using the Advanced Encryption Standard (AES) algorithm, as discussed in Section 1.1.1. These systems exhibit IL through two primary side channel types: **hardware (local)** side channels, such as power consumption and electromagnetic radiations (EMR), which are susceptible to template SCAs for secret key extraction [79]; and **software (remote)** side channels in network traffic, where processing time and error code responses can lead to Bleichenbacher SCAs [80, 81].

Most research in this domain has focused on performing the template and Bleichenbacher SCAs, with recent studies exploring ML and DL approaches for conducting these attacks, as discussed in Section 1.1.2. The channel capacity from the information theory defines MI as an essential measure for quantifying and assessing ILs of the communication channel of cryptographic systems. To establish the comprehensive framework, I refer to literature on template SCA on the AES-encrypted systems linking the MI with ML algorithm loss functions and studies that connect MI with GE (rank) of the secret key of the system under attack, as outlined in Section 1.1.3.

1.1.1. Information Leakage Detection

Information leakage detection (ILD) in cryptographic systems remains crucial for protecting sensitive data since they are vulnerable towards the SCAs due to the presence of **software (remote)** and **hardware (local)** side channels. In terms of OpenSSL TLS servers, most approaches proposed to detect specific **software (remote)** side channels or vulnerabilities are based on using statistical methods or manual analysis of the side information. In the case of template SCAs for secret key extraction using **hardware (local)** side channels, such as power consumption and EMR, typically employ statistical techniques like Gaussian mixture model (GMM). Recent advances have explored ML and DL methods for these tasks, including automated NAS-based solutions for template SCAs, which further inspired the automated approach to perform ILD on AES-encrypted systems.

OpenSSL TLS Servers Remote side channel vulnerabilities in OpenSSL TLS servers have been extensively analyzed to assess their impact, with recent studies performing large-scale evaluations on widely used servers. Notable analyses include the detection of side channel vulnerabilities in ROBOT [21], RACCOON [127], cipher block changing (CBC) padding oracle attacks [128], POODLE [134], MARVIN [102], and others. These analyses generally rely on

sending test messages (vectors) to servers and observing side channel indicators in server responses. However, such methods are prone to false positives due to network instability or variable server behavior, as a single broken Transmission Control Protocol (TCP) connection or timeout can lead to incorrect side channel detection and false negatives in case of novel vulnerabilities or side channels going undetected.

Merget et al. (2019) [128] addressed this challenge by implementing re-scans on vulnerable servers and applying robust statistical tests to improve accuracy. These tests were later integrated into popular Transport Layer Security (TLS) scanning tools, such as SSLlabs[1] and testssl.sh[2], enabling them to cover a wide range of *specific* and *known* vulnerabilities. For example, statistical methods such as FET and chi-squared tests were applied by Böck et al. (2018) [21] to detect Bleichenbacher side channels in the ROBOT attacks. In contrast, TLS-Attacker [108] identified side channels resulting from deviations in TLS message handling and TCP connection state. Despite their effectiveness, these tools rely on explicitly defined vulnerabilities, leaving them unable to detect new side channels that may arise from unforeseen behaviors in TLS implementations or the underlying TCP stack, which are often not analyzed. Tools like TLS-Attacker[3] leverage flexible TLS clients to manipulate padded messages and deviate from standard handshake routines, as seen in the ROBOT [21] or MARVIN [102] attacks. While effective in locating specific side channels, such methods cannot easily combine multiple behavior features, limiting comprehensive detection.

Beyond targeted side channel detection, fuzzing has proven to be a valuable complementary method [174, 195]. Walz and Sikora (2020) [195] explore differential fuzzing, generating a wide array of primarily valid, yet diverse, TLS handshake messages to expose improper server responses, enhancing detection coverage in complex TLS implementations. On the other hand, Ruiter and Poll (2015) [174] introduce protocol state fuzzing, where inferred state machines of TLS implementations reveal unexpected server behaviors or state transitions

[1] `https://www.ssllabs.com/ssltest/`
[2] `https://testssl.sh/`
[3] `https://github.com/TLS-attacker/TLS-Attacker`

that could imply potential side channel risks. While not exhaustive for detecting all ILs (side channels), both fuzzing approaches contribute significantly by identifying hidden, unintended states and code paths that might otherwise lead to exploitable vulnerabilities, thus mitigating potential IL at the protocol level.

A recent approach by Moos et al. (2021) [136] uses a DL classifier for detecting processing time-based ILs in AES-encrypted systems and leverages behavioral features to detect arbitrary leakages. Further, Siavoshani et al. (2023) [180] demonstrated ML-based fingerprinting across 70 domains to identify TLS protocol-specific side channels, highlighting the potential for AutoML approaches to perform ILD or side channel detection in systems vulnerable to SCAs. While effective, their reliance on balanced datasets limits applicability in imbalanced scenarios.

AES-encrypted Systems ILD in AES-encrypted systems provide critical insight into how much information about the secret key can be inferred from side channel traces during the encryption process. As a widely used standard algorithm, AES-encrypted systems remain a crucial target for SCAs that exploit physical leakages like power consumption or EMR to extract the secret key of the system. Given the complexity and multi-round structure of AES encryption, leakage can occur at various stages, making it an ideal subject for ILD approaches.

Additional approaches have been explored using statistical and dynamic side channel behavior analysis for detecting ILs [54, 34, 28]. Notably, He et al. (2017) [82] developed a method for hardware Trojan (HT) detection that leverages EMR side channel spectrum modeling, eliminating the need for golden chips and offering resilience to process variation, making it highly adaptable to varied manufacturing conditions. Similarly, a brain-inspired approach using hierarchical temporal memory (HTM) by Faezi et al. (2021) [54] achieved a high detection accuracy by naturally adapting to side channel changes across an IC's life cycle, bypassing the necessity for golden chip references. These

techniques showcase the potential of current side channel-specific methods but lack generalizability to evolving threats. In contrast, ML-based approaches offer flexibility, adaptability, and automation, making them especially suited for evolving threats in template SCAs on AES-encrypted systems.

While MI estimation is frequently employed for quantifying and detecting ILs, challenges include managing high-dimensional AES traces and capturing non-linear relations between the side information and the secret key bytes. Various methods, including discretization, show slow convergence and susceptibility to noise, particularly in high-dimensional noisy datasets [35]. Recently, DL models have been increasingly applied to perform SCA and detect ILs in AES-encrypted systems [136, 124, 39, 40]. For instance, MINE leverages MI in high-dimensional spaces but often requires extensive training (up to 200 000 epochs), making side-channel analysis (ILD) and SCA computationally intensive [39, 40]. Moreover, MINE faces slow convergence and is sensitive to architecture tuning, particularly in imbalanced datasets [40]. Other models, like Deep Learning Leakage Assessment (DL-LA) proposed by Moos et al. (2021) [136], eliminate the need for pre-processing steps such as trace alignment and multivariate leakage modeling, proving effective in non-profiled SCAs. In this field, the most relevant literature tends to use ML or DL for performing SCAs, rather than focusing on early detection of ILs to prevent them [86, 40]. However, both MINE and DL-LA rely on multi-layer perceptrons (MLPs), which are often outperformed by CNNs in exploiting side channel leakages to perform the SCAs [206, 200, 143, 25]. Notably, current DL-based methods detect side channels and protect systems from SCAs at both the algorithmic and hardware levels [137, 136, 39]. Though DL models have shown effectiveness in detecting ILs by analyzing model accuracy on system data, they struggle with imbalanced, noisy real-world datasets, which can lead to missing novel ILs (false negatives) or detecting non-existent ILs (false positives) [136, 207, 148].

Current Automated ILDs Recent automated solutions for ILD provide valuable advances but remain limited in addressing the full scope of assessing the side channel leakages [194, 159, 9]. Remmerswaal et al. (2024) [159] proposed **AutoPOI**, a framework dedicated to **automated feature selection** for identifying high-leakage points, optimizing template attacks by selecting relevant Points of Interest (POIs) or root cause of the IL. However, it focuses on feature extraction rather than independent detection of IL in the system, which limits its applicability in scenarios requiring comprehensive leakage analysis. Similarly, Walters and Kedaigle (2014) [194] introduced **SLEAK**, which estimates **mutual information (MI)** to identify leakage POIs. However, it assumes independence across intermediate secret information, focusing on **univariate analysis**, which may compromise accuracy when addressing complex multivariate leakages and interdependent data commonly observed in real-world side channel traces. Another tool, **ABBY** by Bazangani et al. (2024) [9], automates **IL modeling** at the microarchitectural layer to simulate side channel behavior rather than performing direct detection of active leakage POIs. Its focus on simulation limits its utility for immediate or complex ILD, especially in dynamic systems with variable countermeasures. Collectively, these tools exhibit strengths in specific areas like feature selection, MI estimation, and simulation, yet they lack the robustness needed for a complete, end-to-end ILD solution that can adapt to complex, multivariate side channel leakages across diverse cryptographic environments. This motivates the need for a comprehensive, end-to-end automated approach for performing ILD that can accommodate multivariate observable information.

1.1.2. Side-channel Attacks

Most of the **software (remote)** SCAs on OpenSSL TLS servers are proposed based on using statistical methods or manual observations of the side information. While the template SCAs for the secret key extraction using **hardware (local)** side channels, such as power consumption and EMR, are performed using statistical techniques like GMM, with recent studies exploring ML and

DL approaches for conducting these attacks. Interestingly, recently, some automated NAS based solutions have also been proposed to perform the template SCAs for recovering the secret key using **hardware (local)** side channels.

OpenSSL TLS Servers Bleichenbacher's attack introduced in 1998 by Bleichenbacher (1998) [20], is a foundational padding oracle SCA targeting cryptographic protocols, whose evolution, history, and similar SCAs are detailed in Section 2.3.2. This attack exploits discrepancies in server responses, such as error codes or processing times, to reveal sensitive information about Rivest–Shamir–Adleman (RSA)-encrypted ciphertexts.

Numerous adaptations of this SCA have emerged, leveraging side channels in protocols like OpenSSL TLS to expose system vulnerabilities using RSA key exchanges. Most recent SCAs on cryptographic OpenSSL TLS protocols bypass algorithmic security by exploiting observable *side channel* information, as demonstrated in DROWN [5], ROBOT [21], RACCOON [127], POODLE [134], MARVIN [102], and others. Padding oracle attacks, including those by Bleichenbacher (1998) [20], Manger (2001) [122], and Vaudenay (2002) [192], have revealed several vulnerabilities in TLS servers [107, 96, 6, 130, 209, 58, 172].

Notably, the ROBOT attack by Böck et al. (2018) [21] uncovered padding oracle vulnerabilities by exploiting subtle differences in TCP session handling, revealing padding oracles in unexpected forms, including variations in TCP termination, regardless of encryption layers. Some adaptations, including Klima's "bad version oracle" Klíma et al. (2003) [107] and more recent improvements, demonstrate the persistence of these vulnerabilities in widely used cryptographic libraries. These vulnerabilities extended across significant products, including Cisco, Citrix, F5, Symantec, Cavium, and Facebook's custom TLS server, highlighting the diversity of padding oracle threats in open-source and proprietary implementations. They even demonstrated a forgery of a valid digital signature using Facebook's RSA certificate based on a padding oracle provided by Facebook's and Cisco's custom TLS server implementation.

Noisy side channels pose challenges for accurate data extraction in SCAs. Yet, techniques like change detection, coding theory, and repeated queries effectively manage errors such as false positives and false negatives [66, 78]. Attacks like PREDATOR and the Marvin SCA leverage precise monitoring and optimizations to exploit noisy conditions, highlighting the potential for ML and DL to offer robust, noise-resistant detection for side channel vulnerabilities [202, 102]. These findings illustrate that padding oracles can emerge in unexpected forms, such as differences in TCP session handling, without direct cryptographic key access, highlighting the need for automated techniques to detect side channels beyond traditional network trace analysis approaches.

Historically, padding oracle SCAs required manual analysis by expert security researchers, who carefully scrutinized individual TLS implementations to identify unique side channel responses. This approach, however, is labor-intensive and needs to scale better as TLS implementations continue to grow over time. Even for experts, detecting unexpected side channels, such as those exploited in the ROBOT attack [21], presents significant challenges, underscoring the pressing need for generalized ML-based to detect side channels beyond typical error code responses in alert messages in cryptographic protocols.

Recent work by Siavoshani et al. (2023) [180] leverages interpretability of the ML models for fingerprinting-based SCAs, using data from over 70 domains to reveal protocol-specific side channels. Although designed primarily for attacks, these ML fingerprinting methods demonstrate the potential for ML-based IL detection, supporting the use of automated ML approaches for side channel identification or performing ILD.

AES-encrypted Systems Attacking cryptographic implementations through hardware side channel emissions, via power consumption and EMR date back to intelligence community operations in the 1950s [109]. The breakthrough in academic research was differential power analysis (DPA), introduced by Kocher et al. (1999) [109], which uses statistical analysis of power consumption or EMR from cryptographic devices to reveal secret keys. This method uses extensive

trace measurements collected over time, which are then matched to possible computations of the executed function using statistical methods to reveal the secret key. While effective, this approach demands extensive observations, such as en- or descriptions, combined with measurements of the target device's power consumption or expectation-maximizations (EMs), limiting real-world applicability. If a device sufficiently similar to the target device can be obtained, for example, by buying a second copy of the device, its profile (a model of its leakage) can be created by observing numerous cryptographic operations with known keys and plaintexts. In the attack phase, fewer measurements must be obtained from the target device, which is subsequently matched with this leakage model. Template SCAs, introduced by Chari et al. (2003) [27], addresses this limitation by creating a leakage model on a similar device with known keys and plaintexts, reducing the number of measurements needed from the target device.

The potential correlation of the secret keys to side channel traces opened new opportunities for ML-based methods, capable of handling noise and misalignment in traces, with early applications achieving notable success [92]. Soon, the ML models grew more sophisticated, and the SCAs became more successful, capable of breaking devices explicitly hardened against SCAs after observing only a handful of attack traces [92, 115, 147, 114, 88, 72, 23, 150]. Sophisticated techniques using CNNs and MLPs have proven powerful for SCA, making DL especially promising, as these models can approximate continuous functions under the universal approximation assumption [41]. However, effective DL models rely on well-chosen architectures with optimal hyperparameters, often requiring labor-intensive trial and error to balance performance and parameter costs [206, 200, 143]. Designing an appropriate architecture can be more of an art than a science, prompting a wave of experimentation with different architectures, both created from scratch and existing ones taken from image classification tasks [14]. In order to alleviate this issue, Perin et al. (2020) [143] proposed using ensembles of multiple networks and aggregating their predictions. While this approach improves the generalization properties of existing CNNs architectures for template SCAs, it also increases the computational

cost and the number of trainable parameters of the model without addressing the underlying issue of architecture design. At the same time, it does not guarantee a successful attack on a new device, e.g., CHES CTF dataset [143]. More recently, Cristiani et al. (2023) [40] introduced a framework for Neural Estimated Mutual Information Analysis (NEMIA), a new attack leveraging DL and information theory, marking the first unsupervised attack that incorporates MI estimated using MINE approach by Belghazi et al. (2018) [12] and outperforming traditional unsupervised SCAs in low-information contexts.

Tuning ML models with appropriate hyperparameter producing the optimal learned model have been shown to outperform classical template SCAs [149, 114, 206, 200]. This limitation drove towards optimizing neural architectures using the NAS approaches, which explore suitable CNN or MLP architectures for performing SCAs, offering promise for improved adaptability and performance in cryptographic vulnerability analysis [201, 169, 1].

Current Automated SCAs In the **hardware (local)** SCA domain, manually crafting optimal attack model architectures has prompted research on automating this process. Recent work by Wu et al. (2024) [201] and Rijsdijk et al. (2021) [169] introduces a white-box approach for automating architecture creation in SCAs, applying NAS with RANDOM and BAYESIAN search strategies and reinforcement learning for architecture optimization. These methods show promising results on datasets like ASCAD and CHES CTF but rely on labeled attack data, raising concerns of overfitting and optimistic performance estimates that may not generalize to unseen data [14].

The Q-function in these models assesses metrics such as GE and the required traces to predict a secret key byte. However, reliance on the attack dataset for hyperparameter optimization (HPO) can lead to data-snooping, compromising generalization [169]. Additionally, using the same dataset for both optimization and testing inflates performance estimates, which poses limitations for black-box scenarios where attack datasets are unlabeled and secret key remains to be discovered [201, 169].

An alternative black-box approach, **InfoNEAT** by Acharya et al. (2023) [1], concurrently advances NAS by evolving multiple architectures and hyperparameters in parallel, using OVR classification for each key byte and employing information-theoretic metrics for SCA. While suitable for **hardware (local)** side channels, InfoNEAT's results cannot be directly compared with single-architecture approaches as it involves distinct scoring for each possible key byte value, reflecting a different model design paradigm.

1.1.3. Foundational Theory

Current theoretical frameworks proposed utilizing MI from information theory to quantify IL in AES-encrypted system, vulnerable towards the **hardware (physical)** SCAs to reveal secret key.

The unified framework by Standaert et al. (2009) [184] bridges the gap between MI used to quantify IL in AES-encrypted system and cryptographic metrics such as GE (key rank) and success rate (SR) (probability of discovering the secret key by adversary), facilitating evaluation of side channel for the secret key recovery attacks. This framework enables a fair assessment of security vulnerabilities in cryptographic devices, identifying optimal leakage exploitation techniques for practical scenarios. Further, for higher-order SCAs approximate score distributed using the multivariate normal (MVN), enabling accurate success rate estimation even when attacks on a complete secret key are infeasible, highlighting the usage of MI and SR in assessing countermeasure effectiveness [118]. Whitnall and Oswald, Chérisey et al. (2011, 2019) [198, 31] proposed that MI-based ILD approaches can excel in noisy scenarios, further affirming MI's utility in robust leakage evaluation, with Chérisey et al. (2019) [31] additionally providing a link between the MI and number of queries or traces required by an adversary to reveal the secret key, i.e.the TST. Additionally, Béguinot et al. (2022) [10] establish a bound on GE relative to SR, while Chérisey et al. (2019) [31] derive an upper bound on MI in terms of SR, indicating that more significant side channel information enhances the adver-

sary's likelihood of correctly guessing the key. Several works further relate MI to GE by bounding GE with conditional entropy, defined as the total entropy of the secret key ($\lg(M)$) minus the MI [123, 31, 125, 11]. Collectively, these frameworks establish MI, GE, SR and as foundational metrics for assessing and mitigating side channel risks in cryptographic systems, with TST providing insight on the ease with which the SCA can be performed by an adversary. In foundational work, Fano (1961) [56] and Hellman and Raviv (1970) [85] introduced bounds on the conditional entropy (lg(#classes) - MI) in terms of the error probability for the Bayes predictor (Bayes error rate). This work provides an initial motivation for using ML based approaches for assessing side channel risks in cryptographic systems and to perform SCAs on them [136, 86]. These bounds laid the groundwork for using MI as a robust metric in performing ILD, allowing security estimations through minimal error probabilities in guessing secret values.

1.2. Outline and Impact

In Section 1.2.1, I outline the thesis and provide co-author contribution statements for the main papers supporting this work in Section 1.2.3. Additionally, I discuss commercial integration and potential industry applications in Section 1.2.4.

1.2.1. Thesis Outline

This thesis is organized into chapters covering distinct aspects of the ILD framework, from foundational concepts to practical applications and empirical evaluations.

Fundamentals Chapter 2 provides the essential theoretical background for developing the ILD framework. In Section 2.1, I introduce fundamental concepts from information theory, such as Rényi entropy and MI, which form the basis for detecting and quantifying information leakage. Section 2.2 presents statistical learning theory, linking classification performance with MI and the conditions under which IL may occur. Finally, Section 2.4 details the statistical tests used in ILD, including FET, PTT, OTT, and Holm-Bonferroni correction, while Section 2.3 describes relevant SCAs, such as Bleichenbacher and template attacks, and Section 2.2.4 discusses AutoML approaches like AutoGluon and TabPFN used to detect and counteract these attacks. The foundational background in this chapter is partially published in Gupta et al. (2024) [81] and Gupta et al. (2023) [79].

Information Leakage Detection In this chapter, I develop the ILD framework by linking Bayes predictor performance with MI, drawing from information theory and statistical learning concepts in Section 2.1 and Section 2.2, respectively. I introduce a generalized LAS measure in Section 3.2.1 to quantify IL and outline two MI-based ILD approaches in Section 3.3 based on Bayes predictor accuracy and Log-Loss. This chapter also introduces MI estimation methods such as Mid-Point and Log-Loss, comparing three state-of-the-art approaches for IL detection in OpenSSL TLS servers, discussed further in Chapter 5. I also propose a specialized GE-based LAS measure for AES-encrypted systems, as detailed in Section 3.2.2, which provides insight into side channel vulnerabilities, expanded in Chapter 6. This chapter is based on the work published in Gupta et al. (2024) [81] and Gupta et al. (2022) [80].

Mutual Information Estimation This chapter assesses the MI estimation methods against state-of-the-art methods using synthetic datasets generated by MVN distributions. In Section 4.1, I describe the process of generating synthetic datasets for testing, while Section 4.2 details the empirical setup and normalized mean absolute error (NMAE) as a performance metric. Results are

summarized in Section 4.3.1, with additional insights into generalization capabilities across various dimensions and noise levels, as discussed in Section 4.3.2 and further analyzed in Appendix A.4. The findings demonstrate that the proposed approaches outperform the existing baselines. This chapter is based on the work published in Gupta et al. (2024) [81].

Automating ILD in OpenSSL TLS Servers This chapter evaluates ILD approaches for detecting ILs through time delays and alert/error messages to counter Bleichenbacher SCAs on OpenSSL TLS servers. Section 5.2 outlines the empirical setup and provides an overview of the ILD datasets, which are generated in Section 5.1. Results confirm that the proposed ILD approaches outperform the current state-of-the-art, presented in Section 5.3. This chapter is based on the work published in Gupta et al. (2024) [81].

Automating ILD in AES-encrypted Systems In this chapter, I present an automated ILD framework that uses GE and TST values to detect leakage in AES-encrypted systems. The setup for analyzing optimal parameters, including NAS search strategies and input feature design, is covered in Section 6.2. Datasets and implementation details are presented in Section 6.2.2 and Appendix A.1.3. My results on parameter selection, IL assessment in terms of VS, and efficiency in terms of TST are analyzed in Section 6.3. This chapter is based on the work published in Gupta et al. (2023) [79].

Summary and Future Directions The final chapter provides an overall conclusion of the thesis, summarizing key findings and contributions in Section 7.1. Future research directions are outlined in Section 7.2, emphasizing improving ILD frameworks and expanding their applications to other cryptographic systems and real-world security challenges. The conclusion and outlook are partially published in Gupta et al. (2024) [81] and Gupta et al. (2023) [79].

1.2.2. Research Focus

Though ML methods have been applied to evaluate side channel risks and conduct ILD in cryptographic systems, there has been no comprehensive framework linking the Bayes predictor from statistical learning theory and MI from information theory for ILD. Current approaches primarily focus on estimating MI with DL models but have not demonstrated how MI can be directly evaluated using the log-loss of Bayes predictor, specifically applicable for the setting of AES-encrypted producing classification dataset [12, 39, 40, 158]. Additionally, foundational metrics like GE, SR, and TST assess an adversary's efficiency in performing SCAs but do not quantify a system's susceptibility to SCAs in expectation. This work addresses these gaps by establishing a unified theoretical framework that leverages the relationship between MI and Bayes predictor performance, quantifying IL through leakage assessment score (LAS) and formalizing its existence conditions in a system. For detecting ILs in AES-encrypted systems, I propose VS metric, which is based on LAS defined using GE (rank of the secret key byte). The VS combined with TST provides insights on the relative ease of compromising systems through SCAs. I also propose two MI estimation methods that approximate the Bayes predictor's performance using AutoML tools, demonstrating their effectiveness through empirical validation.

Current ILD and SCA methods predominantly rely on ML or DL with fixed architectures and predefined hyperparameters [39, 40, 86, 142, 136], limiting adaptability across diverse side channel data and often resulting in suboptimal performance, compromising generalizability and inconsistent effectiveness of the ILD approach. To address this, recent automated solutions utilizing AutoML approaches for optimized hyperparameters and architectures have been proposed [194, 159, 9]. Further, approaches using NAS (a subset of AutoML) have shown potential for adapting CNN architectures specifically for the side information in form of EMR and power consumption to perform the template SCAs in AES-encrypted systems [201, 169, 1]. I propose leveraging NAS to enable more robust ILD for AES-encrypted systems, while AutoML

tools such as TabPFN and AutoGluon facilitating fully automated detection of ILs in OpenSSL TLS servers, improving robustness and reducing the need for manual configuration.

The current work is limited to detecting ILs in OpenSSL TLS servers and AES-encrypted systems, producing balanced classification dataset [136, 49]. I propose incorporating weighted loss functions within AutoMLs pipeline, integrating statistical tests like FET and PTT to account for class imbalance [80, 81]. To mitigate noise, I aggregate resulting p-values from an ensemble of the top-10 performing AutoMLs pipelines using the Holm-Bonferroni correction to provide confidence in detection accuracy and produce a robust solution towards noise [81]. I propose employing a weighted loss function to perform ILD and side channel vulnerability analysis in AES-encrypted systems by running multiple NAS trials across 10 deciles of attack datasets, providing a robust solution to noise. Subsequently, I aggregate the GE estimates to generate the VS, which, combined with TST, assesses the relative ease of compromising systems through SCAs. Previous NAS research in SCA has focused on Bayesian and Random search strategies, which can be computationally inefficient [201, 160]. I enhance this by evaluating Random, Greedy, Hyperband, and Bayesian search methods through AutoKeras to optimize architectures more systematically. Additionally, I extend NAS based SCA on AES-encrypted systems by incorporating search spaces using the 2-D CNN models already proposed to break the post-quantum key-exchange (PQKE) protocols for improved attack performance [103, 87].

Contributions

I address the limitations in current related work and unanswered research questions through the following contributions

- I establish a comprehensive theoretical framework leveraging the connection between MI and the performance of the Bayes predictor to quantify IL using leakage assessment score (LAS) and formalize its existence conditions in a system.
- To induce the Bayes predictor, I propose leveraging AutoML tools and NAS approaches, which incorporate consistent learners such as ML ensemble methods (random forest classifier (RF), gradient boosting machine (GBM)) and DL architectures like MLPs and CNNs [18, 132]. I also propose two MI estimation methods by approximating the Bayes predictor induced using AutoML tools and demonstrate their effectiveness through a rigorous empirical evaluation on synthetic datasets generated using MVN distribution.
- As MI estimation becomes increasingly challenging with 256 possible classes, representing the secret key byte, detecting ILs in AES-encrypted systems benefits from the proposed VS metric, which is based on LAS using the GE (secret key byte rank) of the system. The VS, combined with the TST metric, offers more profound insights into the efficiency of possible SCAs and how easily the given systems can be compromised.
- Using a cut-off on estimated MI through the OTT, I devise a MI-based technique for performing ILD in systems generating classification datasets. I also propose classification-based approaches, which utilize the FET on the confusion matrix and PTT on accuracy estimates of the AutoML tools, to detect IL.
- Furthermore, I propose using the Holm-Bonferroni correction on multiple models' estimates to enhance IL detection confidence by making it robust against noise and variations in AutoML pipelines' quality.
- I conduct an extensive empirical study, comparing the ILD methods against state-of-the-art approaches for detecting sensitive information leaked via processing time and error code in the network trace side information to counter Bleichenbacher's SCAs.

- Inspired by automated SCAs, I propose a black-box NAS-based approach for detecting ILs in AES-encrypted systems. This method performs multiple independent SCAs to compute a VS, yielding more reliable GE performance estimates.
- I expand the previous NAS-based experiments to perform SCAs into a large-scale parameter study, investigating the impact of the search strategy and the input feature shapes. I consider four different search strategies, including Greedy and Hyperband. Additionally, I explore the transformation of one-dimensional traces into two-dimensional inputs for template SCAs, leveraging NAS with 1-D and 2-D CNN architectures, inspired by image classification models like VGGNet and Inceptionv3, to improve the efficiency. The evaluation is performed on 10 publicly available reference datasets in identity (ID) leakage model.
- I also conduct a performance comparison between the CNN architectures obtained from the NAS approach and the state-of-the-art fixed architectures for performing SCA on these datasets proposed by Benadjila et al. (2020) [14] and Zaid et al. (2019) [206].

1.2.3. Co-author Contribution Statements

As outlined in the respective chapters, portions of this thesis were published in workshops, conferences, and journals during my Ph.D.studies.The publications are organized according to the thesis chapters that build upon them. This thesis builds upon the work published in the "18th International Conference on Availability, Reliability, and Security" by Gupta et al. (2023) [79] and the journal paper under submission [81], which expands on work initially presented at the "14th International Conference on Agents and Artificial Intelligence" by Gupta et al. (2022) [80]. Additionally, I used some parts for describing the fundamentals of Bleichenbacher's SCA of the first workshop paper published in the "14th ACM Workshop on Artificial Intelligence and Security" by Drees et

al. (2021) [49]. I had the privilege of collaborating with outstanding colleagues on these works, and in this section, I detail my specific contributions alongside those of my co-authors.

Information Leakage Detection through Approximate Bayes-optimal Prediction

As the first author, I established a theoretical framework linking the Bayes predictor performance with MI for ILD. I introduced the Mid-Point MI estimation approach and, in collaboration with Eyke Hüllermeier, developed the Log-Loss MI estimation method to address the limitations of the Mid-Point approach. The contributions included exploring calibration techniques for MI estimation, proposing baseline models like GMM, MINE, and probabilitiy-corrected softmax (PC-softmax), and enhancing the robustness of the GMM and MINE approaches for high-dimensional data. I also designed the experimental setup, generated synthetic datasets, and proposed statistical tests and AutoML pipelines with Holm-Bonferroni correction to ensure reliable ILD outcomes. Marcel Wever supported the research by exploring recent AutoML tools, refining the result presentation, and guiding the structuring of the paper. Eyke Hüllermeier contributed by defining key concepts from the information and statistical learning theory and suggesting overlapping MVN distributions could introduce noise in synthetic datasets.

Automated side channel Attacks using Black-Box Neural Architecture Search

As the first author, I implemented the attack methodology on the Noctua2 cluster, defining the search space, selecting baseline models, and designing input-reshaping strategies for effective model training. I developed search strategies for NAS, analyzed attack success rates across models, and studied NAS-based approaches and their limitations within ML. I compiled the complete methodology for black-box NAS-based methodology for testing the AES encryption

systems vulnerable towards the template SCAs. Eyke Hüllermeier provided formal guidance on NAS applications, helped shape the theoretical framework for attack convergence, and refined the paper's analysis and structure. Jan Peter Drees and I jointly collected the datasets, developed the black-box SCA methodology, and performed the result analysis and interpretation.

Automated Information Leakage Detection: A New Method Combining Machine Learning and Hypothesis Testing with an Application to side channel Detection in Cryptographic Protocols

As the first author, I collaborated with Arunselvan Ramaswamy, Eyke Hüllermeier, Claudia Priesterjahn, and Tibor Jager to publish this work at the 14th International Conference on Agents and Artificial Intelligence. My contributions included developing the relationship between IL, the Bayes predictor accuracy, proposing and implementing PTT and FET for ILD, generating synthetic datasets, and acquiring the results. Jointly with Jan Peter Drees, we proposed using the Holm-Bonferroni correction and conducted result analysis on real-world OpenSSL TLS datasets.

Automated Detection of Side Channels in Cryptographic Protocols: DROWN the ROBOTs!

As the first author, Jan Peter Drees led this joint work with myself, Eyke Hüllermeier, Tibor Jager, Alexander Konze, Claudia Priesterjahn, Arunselvan Ramaswamy, and Juraj Somorovsky. He also contributed to the modular framework for analysis of TLS clients, implemented the AutoSCA tool, designed the feature extraction process, conducted experiments, and analyzed results. My contributions included designing and implementing the ML module, analyzing feature importance, and collaborating on result analysis and discussions.

Additional Publications

During my tenure with the research group (ISML, Paderborn University or KIML, LMU Munich) under Prof.Dr.Eyke Hüllermieier, I contributed to additional publications in preference learning and outlier detection as

- **Context-Dependent Choice Functions**: Pfannschmidt et al. (2022) [145] addresses the problem of learning choice functions under context dependence, modeling how user preferences shift based on available options, using context-dependent utility functions.
- **Deep Architectures for Context-Dependent Ranking**: Pfannschmidt et al. (2018) [146] introduces deep neural network models to capture context-dependent rankings, adjusting predicted orderings based on other available objects in the set.
- **Pairwise vs.Pointwise Ranking**: Melnikov et al. (2016) [126] provides a comparative study on pairwise versus pointwise ranking techniques, particularly focusing on object ranking in preference learning applications.
- **Outlier Detection for Semi-Supervised Datasets**: Schubert et al. (2023) [176] explores meta-learning approaches to automate the selection of optimal outlier detectors, using metrics for detection performance when only normal data is available.

1.2.4. Thesis Impact

The contributions made within this thesis hold the potential for significant impact in the field of cryptographic system testing, particularly for OpenSSL TLS servers and AES-encrypted devices, such as smart cards, hardware wallets, and more.

Open Source Tool

The AutoSCA tool[4] provides a robust, automated solution for detecting side channels in cryptographic protocols by leveraging ML techniques to analyze network traffic and identify vulnerabilities. This tool implements the solution of using the FET to detect IL in a system [156, 49]. The tool and its data[5] are made publicly available as open-source on GitHub, promoting accessible and transparent security testing in cryptographic systems.

As illustrated in Figure 1.1, the tool has a structured pipeline that executes sequentially to ensure effective side channel detection. The stages of this tool, as described in [156, 49], include:

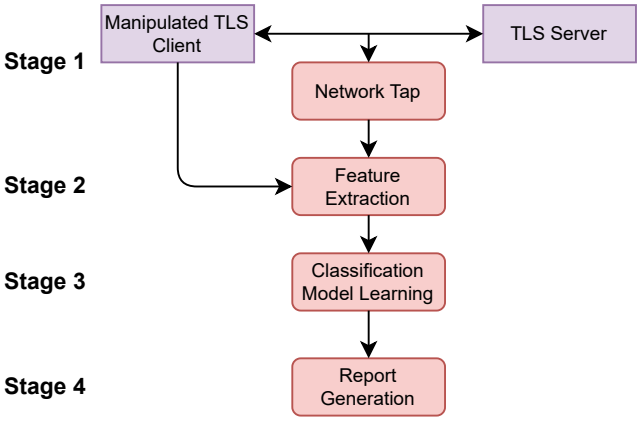


Figure 1.1.: Components of the AutoSCA tool [49]

1. **Manipulated TLS Client**: This component initiates testing by connecting to the TLS server under analysis and executing a series of requests with controlled padding manipulations, enabling the capture of subtle variations in the server's responses.
2. **Feature Extraction**: Following the collection of raw data from TLS client-server interactions, this stage converts the data into a ML-compatible format, extracting essential features that indicate potential IL.

[4] https://github.com/ITSC-Group/autosca-tool
[5] https://github.com/ITSC-Group/autosca-data

3. **Classification Model Learning**: Various ML classifiers are trained on the extracted dataset to detect side channels. The ensemble of classifiers enhances detection accuracy by leveraging the strengths of different algorithms.

4. **Report Generation**: Finally, the tool generates a comprehensive report summarizing findings, including a vulnerability assessment based on the Holm-Bonferroni statistical test and feature importance analysis. This output assists developers in pinpointing specific vulnerabilities and implementing necessary fixes.

One particularly effective solution within this tool provides the option of using FET (FET-Median, FET-Mean) and PTT (PTT-Majority) combined with the Holm-Bonferroni correction to detect IL in a system, as described in Section 3.2.2 and proven effective in Chapter 5. This method is especially effective for smaller datasets, enhancing detection reliability by minimizing statistical outliers. The capability of this tool to handle complex side channel vulnerabilities makes it an invaluable asset for developers working to secure cryptographic protocols and systems. In the future, I would like to update this tool to integrate AutoML based solutions with MI based ILD approaches described in Section 3.2.2, to provide more variety of solutions.

Commercial Integration The detection mechanism was integrated into the TLS Server Inspector by achelos GmbH, a commercial tool used to test TLS servers for compliance in sensitive sectors like healthcare[6]. Thanks to the modular design of the AutoSCA tool, achelos GmbH engineers could easily incorporate components like the TLS client and network tap, as shown in Figure 1.1. This integration enables fully automated side channel detection, allowing testers without cryptography expertise to include the Bleichenbacher test seamlessly within a broader security test suite.

[6] https://www.achelos.de/de/services-loesungen/testsuiten/tls-test-tool/

Application to Future Projects

I recently discussed the potential application of the proposed testing solution from Chapter 6, developed from the work in Gupta et al. (2023) [79], with Diebold-Nixdorf employees, who visited my poster during the events like Paderborner Tag der IT-Sicherheit[7]. They highlighted its potential for detecting IL in ATMs and smart cards, supporting the prevention of identity theft and phishing. During a visit to NSUT in Delhi, India, I also discussed with venture capitalist Manusheel Gupta working with companies like Etherbit[8] the potential of applying this solution to enhance security in hardware wallets against IL.

1.3. Notation and Diagram Legend

This section introduces a consistent set of notations used throughout the thesis to maintain clarity, as summarized in Table 1.1, covering key symbols from probability, information theory, and statistical learning theory, used for performing SCA and ILD. Additionally, diagrams follow a standardized visual format, demonstrated in Figure 1.2, to uniformly illustrate workflows and processes, particularly in explaining ILD and MI estimation methods, ensuring ease of understanding through clear, consistent visuals. In this work, I the pair of random variables X and Y with a joint distribution $p_{(X,Y)}(\cdot)$ over $\mathcal{X} \times \mathcal{Y}$, such that X is a continuous d-dimensional real-valued variable ($\mathcal{X} = \mathbb{R}^d$) and Y is a discrete variable with M possible values ($\mathcal{Y} = \{1, \ldots, M\}$). In the field of ILD, this notation applies to cases where X represents observed traces, and Y represents possible outputs. In the following, I simplify the notation by using the same symbols for probability measures and their mass or density functions, relying on context to distinguish between them.The marginals of the joint distribution $p_{(X,Y)}(\cdot)$ are denoted by $p_X(\cdot)$ and $p_Y(\cdot)$, with conditional distributions represented by $p_{Y|X}(\cdot)$ and $p_{X|Y}(\cdot)$. For statistical learning, I

[7] https://www.sicp.de/aktuelle-veranstaltungen/paderborner-tag-der-it-sicherheit
[8] https://www.etherbit.in/collections/wallets

Table 1.1.: The table summarizes key notation utilized throughout the thesis.

Symbol	Meaning
$[n]$	Set of integers $\{1, 2, \ldots, n\}, n \in \mathbb{N}$
$[n]_0$	Set of integers $\{0, 1, \ldots, n-1\}, n \in \mathbb{N}$
$[\![A]\!]$	Indicator function which is 1 if statement A is true and 0 otherwise
	Probability Distribution Functions
$p_{(X,Y)}(\cdot)$, $\hat{p}_{(X,Y)}(\cdot)$	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
	Statistical Learning Theory
X	Input ($\boldsymbol{x} \in \mathbb{R}^d$) random variable (d-dimensional continuous), Observed EMR traces
Y	Output ($y \in [M]$) random variable (discrete), Intermediate S-box output from AES algorithm
$\mathcal{X} \in \mathbb{R}^d$	Input Space, set of $\boldsymbol{x}$ sampled from X
$\mathcal{Y} \in [M]$	Output Space, set of y sampled from Y
$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$	Classification dataset
r	Imbalance in a dataset $\mathcal{D}$
ϵ	Noise in a dataset $\mathcal{D}$
	Information Theory
$I(X;Y)$	MI between X and Y
$H(Y\|X)$	Conditional entropy for Y given X
$H(X)\,, H(Y)$	Entropy for random variable X and Y
$H_2(a) = -(a)\lg(a) - (1-a)\lg(1-a)$	Binary cross-entropy function for $a \in (0, 1)$
$\lg(a), \log(a), \ln(a), a \in \mathbb{R}^+$	Binary (base 2), Decimal (base 10) and Natural (base e) of a
	Information leakage detection (ILD) process
$\mathrm{m}_{\mathrm{ERR}}(g^{bc}), \mathrm{m}_{\mathrm{ERR}}(g^{mc})$	The Bayes error rate and error rate of marginal Bayes predictor
$\delta(\ell_{(\cdot)}) = \ell_{(\cdot)}(g^{mc}) - \ell_{(\cdot)}(g^{bc})$ $\delta(\mathrm{m}_{(\cdot)}) = \lvert\mathrm{m}_{(\cdot)}(g^{mc}) - \mathrm{m}_{(\cdot)}(g^{bc})\rvert$	LAS is the difference in performance of g^{bc} and g^{mc} quantifying IL
$L\,, \mathcal{L}$	ILD function and IL-Dataset
H_0(condition)$\,, H_1$(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
$\alpha = 0.01$	Rejection threshold on H_0 (accept H_1) for statistical tests
K, $\hat{K}$	Device's and Predicted 16-byte Secret Key, represented as $\boldsymbol{k} = (k_0, \ldots, k_{15}), \forall k_i \in \mathcal{K}$
T	16-byte Plaintext random variable (discrete vector) in the space $([256]_0)^{16}$, such that $\boldsymbol{t} = (t_0, \ldots, t_{15}), \forall t_i \in \mathcal{T}$
$\mathcal{T} \in [256]_0$	Plaintext Byte Space, set of bytes from which the plaintext $\boldsymbol{t}$ is sampled
$\mathcal{K} \in [256]_0$	Secret Key-byte Space, where key bytes k are sampled such that $k \in K$ or $\hat{k} \in \hat{K}$

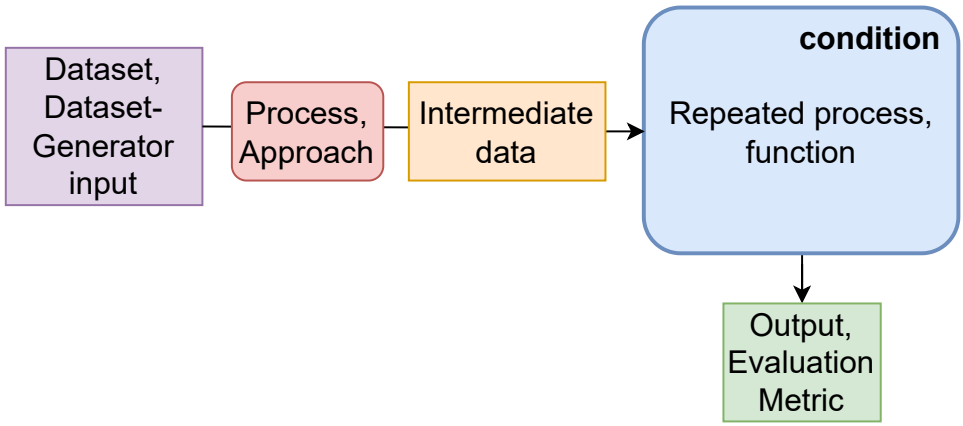


Figure 1.2.: Diagram design format

denote X as the input variable with $\mathcal{X}$ as its space, while Y is the output space variable associated with $\mathcal{Y}$. The dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ contains instances sampled from X and Y, with parameters r and ϵ for class imbalance and noise.Information-theoretic concepts include $I(X;Y)$, representing mutual information, $H(Y|X)$ as conditional entropy, and $H(X)$ or $H(Y)$ for the entropy of each random variable. In the ILD process, LAS quantifies IL through the difference in classifier performance between g^{bc} and g^{mc}. The statistical tests involve null (H_0) and alternate hypotheses (H_1), using a Holm-Bonferroni correction with threshold $\alpha = 0.01$ and cut-off parameter τ used to imply the existence of IL in a system. Finally, device secret keys K and predicted secret key $\hat{K}$ are represented as 16-byte vectors in space $\mathcal{K}$ and the plaintexts denoted by the random variable T and represented by space $\mathcal{T}$, respectively, crucial for explaining the template SCAs.

2. Fundamentals

This chapter provides the foundational background for the framework and automated information leakage detection (ILD) approaches leveraging automated machine learning (AutoML) tools. Section 2.1 covers concepts from information theory, such as Rényi entropy and mutual information (MI), crucial for detecting and quantifying information leakage (IL) in systems. Section 2.2 introduces statistical learning theory, the classification problem, and the Bayes predictor, which are later connected to MI and used to derive the conditions for the occurrence of IL in a system, as discussed in Chapter 3. The AutoML methods used to induce the Bayes predictor and automate ILD to counter template and Bleichenbacher's side-channel attacks (SCAs) are described in Section 2.2.4. Section 2.4 outlines statistical tests used for ILD, while Section 2.3 details Bleichenbacher's and template SCAs, as detecting ILs in systems vulnerable to these attacks is a central focus of this thesis. Notations and diagram legends are introduced in Section 1.3.a This chapter is partially based on Gupta et al. (2024) [81], Gupta et al. (2023) [79] and Drees et al. (2021) [49].

2.1. Information Theory

This section introduces Rényi's entropy, a generalized measure for quantifying information that preserves additivity for independent events [170, 161]. The discussion then transitions to Shannon entropy as a specific case, followed by an explanation of MI, which quantifies IL in cryptographic systems by measuring the channel capacity [81].

2.1.1. Entropy

Rényi (1961) [161] introduced a generalized measure of entropy, called **Rényi's entropy**, which unifies various notions of entropy, including Hartley entropy, Shannon entropy, collision entropy, and min-entropy, through a tunable parameter α. This parameter adjusts the sensitivity of the entropy measure to class probabilities, highlighting different characteristics of a probability distribution.It provides a general framework to quantify information while preserving additivity for independent events, making it a versatile metric for quantifying the information content of a variable [170, 8].

In the following work, the same notation is used for a probability measure and its probability mass (probability mass function (PMF)) or density function (probability density function (PDF)) for simplicity. Typically, capital letters represent the former and lowercase letters the latter; however, lowercase letters are consistently applied throughout, with the context providing clarity regarding the intended meaning.

Rényi entropy can be defined for discrete and continuous random variables [161]. In this work, entropy is calculated using the binary logarithm $\lg(\cdot)$, with *bits* as the unit of information. For a d-dimensional real-valued random variable X is a continuous with PDF $p_X(\cdot)$ with space $\boldsymbol{x} \in \mathcal{X}$, the Rényi's, α-entropy is defined as

$$H_\alpha(X) := \frac{1}{1-\alpha} \lg \left(\int\limits_{\boldsymbol{x}\in\mathcal{X}} p_X(\boldsymbol{x})^\alpha \, d\boldsymbol{x} \right).$$

Generally, in the context of cryptographic systems, leakage is associated with the output discrete random variable Y, representing the discrete secret information with M possible values ($\mathcal{Y} = \{1, \ldots, M\}$). Using the corresponding probabilities $p_Y(y_i) = p_i$, the Rényi α-entropy is defined as

$$H_\alpha(Y) := \frac{1}{1-\alpha} \lg \left(\sum_{y\in\mathcal{Y}} p_Y(y)^\alpha \right), \; \alpha > 0, \; \alpha \neq 1.$$

In case of probabilities of Y being uniformly distributed and occurs when all outcomes are equally likely, i.e., $p_Y(y) = 1/M$, $\forall y \in [M]_0$, all Rényi entropies of the distribution are equal $H_\alpha(Y) = \lg(M)$.

Special Cases

This section describes the different special cases of Rényi α-entropy for discrete random variable Y, relevant to us for IL scenario, for $\alpha \to i, i \in \{0, 1, 2, \infty\}$, namely Hartley, Shannon, Collision and Min-Entropy [161]. With an increasing value of α, the entropy measure increasingly focuses on the most likely events, while as α approaches zero, all events with nonzero probability are weighted more equally, irrespective of their actual probabilities as shown in Figure 2.1.

Hartley Entropy $\alpha \to 0$ Hartley entropy H_0 of Y is the logarithm of the number of possible outcomes and is defined as

$$H_0(Y) := \lg|\{y \in \mathcal{Y} : p_Y(y) > 0\}|,$$

It is essentially the support size of Y, which considers only whether an event can occur, not how likely it is.

Shannon Entropy $\alpha \to 1$ Shannon entropy is expected value of the random variable $\lg {}^1/_{p_Y(\cdot)}$

$$H_1(Y) := -\sum_{y \in \mathcal{Y}} p_Y(y) \lg(p_Y(y)),$$

This is a standard entropy measure used for quantifying the uncertainty or information content in a random variable.

Collision Entropy $\alpha \to 2$ Collision entropy is often used in estimating the likelihood of two randomly chosen events being the same:

$$H_2(Y) := -\lg \sum_{y \in \mathcal{Y}}^{n} p_Y(y)^2,$$

This entropy measure emphasizes repeated occurrences within a distribution, making it relevant in scenarios with significant duplicates or collisions.

Min-Entropy $\alpha \to \infty$ Min-entropy focuses on the most probable event, providing a measure of the tightest bound on the predictability of Y:

$$H_\infty(Y) := \min_{y \in \mathcal{Y}} -\lg(p_Y(y)) = -\max_{y \in \mathcal{Y}} \lg(p_Y(y)) = -\lg \max_{y \in \mathcal{Y}} p_Y(y)$$

Min-entropy is used in randomness extractors and provides a measure of predictability, focusing on the most probable event. It is crucial in the context of IL detection and quantification as it assesses the worst-case scenario, where the adversary's advantage is maximized. In the context of SCAs, it evaluates system security by identifying the most crucial leakage points.

Developing a unified framework for detecting ILs in cryptographic systems highlights Rényi entropy's potential in exploratory diagnostics, offering sharper predictions for two-class scenarios and measuring uncertainty reduction when information is exposed [13]. The relationship between entropy and minimal error probabilities further provides bounds to quantify IL, assessing adversarial success through a channel coding perspective [57]. Additionally, Rényi divergence metrics establish a robust security framework under noisy conditions for cryptographic primitives [155]. Nevertheless, this thesis focuses on (Shannon entropy $\alpha \to 1$) MI estimation methods for quantifying and detecting ILs, ensuring computational feasibility for side-channel analysis.

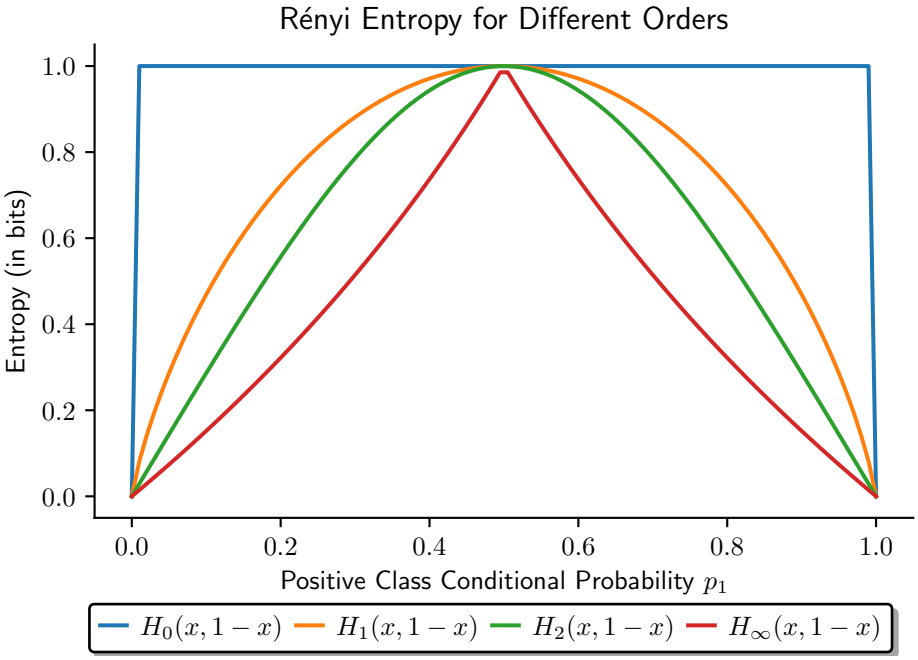


Figure 2.1.: Rényi Entropy for Different Orders [36]

2.1.2. Mutual Information

Mutual information (MI), a fundamental concept in information theory, measures the extent to which knowledge of one random variable informs about another, thereby quantifying their degree of dependence [38]. To define the MI, a pair of random variables X and Y with joint distribution $p_{(X,Y)}(\cdot)$ on space $\mathcal{X} \times \mathcal{Y}$ are considered. Although these random variables can be of any type, it is subsequently assumed that X is a continuous d-dimensional real-valued random variable ($\mathcal{X} = \mathbb{R}^d$) and Y is a discrete random variable with M possible values ($\mathcal{Y} = \{1, \ldots, M\}$) — in the information leakage (IL) scenario, this will be the relevant case for us. The marginals of the joint distribution $p_{(X,Y)}(\cdot)$ is denoted by $p_X(\cdot)$ and $p_Y(\cdot)$, and the conditional distributions by $p_{Y|X}(\cdot)$ and $p_{X|Y}(\cdot)$, respectively. Recall, that the *Shannon* entropy (Rényi's entropy with $\alpha \to 1$) of a discrete random variable Y is defined as

$$H(Y) := -\sum_{y \in \mathcal{Y}} p_Y(y) \cdot \lg(p_Y(y)) \,, \tag{2.1}$$

where $0 \cdot \lg(0) = 0$ by definition; the continuous version — differential entropy — is essentially obtained by replacing summation with integration. The maximum entropy is $\lg(M)$ and occurs when all outcomes are equally likely, indicating complete uncertainty about the outcome of Y, i.e., $p_Y(y) = 1/M$, $\forall y \in [M]_0$. The minimum entropy is 0, indicating complete certainty about Y, and occurs for the case of a Dirac measure (one outcome $y = m$ has a probability $p_Y(m) = 1$, while all others have a probability of 0, i.e., $p_Y(i) = 0\,, \forall i \in [M]_0 \setminus \{m\}$). The conditional entropy of Y given X is defined as

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

According to this definition, conditional entropy measures the residual uncertainty in one random variable given knowledge of the other — it measures the *expected* residual uncertainty, with the expectation taken with respect to the marginal distribution of Y. $H(Y|X)$ is bounded by $H(Y)$, reaching its upper bound when $H(Y \,|\, \boldsymbol{x})$ follows the marginal distribution $p_Y(\cdot)$ for all $\boldsymbol{x} \in \mathcal{X}$, signifying that X does not provide any information about Y. The minimum (conditional) entropy is 0, indicating that X completely determines Y, and again occurs in the case where all conditionals $p_{Y|X}(\cdot \,|\, \boldsymbol{x})$ are Dirac distributions.

MI between a pair of random variables (X, Y) measures the reduction of uncertainty about variable Y by observing variable X, is defined as

$$I(X;Y) := H(Y) - H(Y|X)\,. \tag{2.2}$$

Consequently, its value ranges from 0 to $\min(\{H(X), H(Y)\})$. A value of 0 indicates that X and Y are completely independent, while the maximal value $\min(\{H(X), H(Y)\})$ signifies full dependence [38].

Properties

The MI is a non-negative and symmetric measure, aligning well with the nature of IL, which is also non-negative ($\mathbb{R}^+$). Symmetry ensures MI can be interpreted from both perspectives ($X \to Y$ and $Y \to X$). in Section 3.3.2. These properties, alongside MI's ability to differentiate dependent and independent variables, make it an appropriate measure for quantifying IL, as discussed in Section 3.2.2. Estimation approaches such as the Gaussian mixture model (GMM) baseline and the proposed LOG-LOSS method (c.f. Section 3.3) leverage these properties to distinguish between dependent and independent variables, using eqs. (2.2) and (2.3), respectively.

Non-negativity By plugging and rearranging terms in the expressions for (conditional) entropy in eq. (2.2), one easily 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 independence assumption):

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= \int_{\boldsymbol{x}\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{(X,Y)}(\boldsymbol{x},y) \cdot \lg\left(\frac{p_{(X,Y)}(\boldsymbol{x},y)}{p_X(\boldsymbol{x})\cdot p_Y(y)}\right) d\boldsymbol{x} \qquad (2.3) \\ &= D(p_{(X,Y)}(\boldsymbol{x},y) \| (p_Y(y)\cdot p_X((\boldsymbol{x}))) = D(p_{(X,Y)}(\boldsymbol{x},y) \| q_{(X,Y)}(\boldsymbol{x},y)) \\ &= \mathbb{E}_{(\boldsymbol{x},y)\sim p_{(X,Y)}(\boldsymbol{x},y)}\left[\lg\left(\frac{p_{(X,Y)}(\boldsymbol{x},y)}{q_{(X,Y)}(\boldsymbol{x},y)}\right)\right] \end{aligned}$$

where $q_{(X,Y)}(\cdot)$ represent the product of the marginals and $D(p_{(X,Y)}(\cdot)\|q_{(X,Y)}(\cdot))$, represents the KL divergence between $p_{(X,Y)}(\cdot)$ and $q_{(X,Y)}(\cdot)$, which is also the expectation of $\lg(p_{(X,Y)}(\cdot)/q_{(X,Y)}(\cdot))$ under the joint distribution $p_{(X,Y)}(\boldsymbol{x},y)$.

Since $\lg(t)$, $t \in \mathbb{R}^+$ is concave, the Jensen's inequality $\mathbb{E}_{t\in\mathbb{R}^+}[\lg(t)] \leq \lg(\mathbb{E}_{t\in\mathbb{R}^+}[t])$ is applied to $D(p_{(X,Y)}(\cdot)\|q_{(X,Y)}(\cdot))$ as follows

$$\begin{aligned}
-D(p_{(X,Y)}(\cdot)\|q_{(X,Y)}(\cdot)) &= -\mathbb{E}_{(\boldsymbol{x},y)\sim p_{(X,Y)}(\boldsymbol{x},y)}\left[\lg\left(\frac{p_{(X,Y)}(\boldsymbol{x},y)}{q_{(X,Y)}(\boldsymbol{x},y)}\right)\right] \\
&= \int_{\boldsymbol{x}\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{(X,Y)}(\boldsymbol{x},y) \lg\left(\frac{q_{(X,Y)}(\boldsymbol{x},y)}{p_{(X,Y)}(\boldsymbol{x},y)}\right) d\boldsymbol{x} \\
&\leq \lg\left(\int_{\boldsymbol{x}\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{(X,Y)}(\boldsymbol{x},y)\left(\frac{q_{(X,Y)}(\boldsymbol{x},y)}{p_{(X,Y)}(\boldsymbol{x},y)}\right) d\boldsymbol{x}\right) = 0 \\
&\implies D(p_{(X,Y)}(\cdot)\|q_{(X,Y)}(\cdot)) \geq 0 \implies I(X;Y) \geq 0
\end{aligned}$$

with equality if and only if X and Y are independent, i.e., $p_{(X,Y)}(\boldsymbol{x},y) = q_{(X,Y)}(\boldsymbol{x},y) = p_Y(y)\cdot p_X(\boldsymbol{x})$, $\forall(\boldsymbol{x},y) \in \mathcal{X}\times\mathcal{Y}$. This confirms the non-negativity of the KL divergence and consequently the MI [38].

Symmetry MI is a symmetric measure, indicating that the shared information between X and Y is the same in both directions [38].

$$\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) \\
&= H(Y) + H(X) - H(X,Y) = H(X) + H(Y) - H(Y,X) \\
&= H(X) - H(X\,|\,Y) = I(Y;X)\,.
\end{aligned}$$

2.2. Statistical Learning Theory: Classification Problem

This section revisits fundamental concepts of classification problems and the relevant metrics from statistical learning theory, with notations summarized in Table 1.1. Building on these, the concept of the Bayes predictor and the classification problem is introduced, which estimates (class) conditional probabilities to induce the Bayes predictor. These concepts are essential

for formalizing the conditions for IL occurrence in a system, as discussed in Section 3.2.1. Furthermore, the benchmark AutoML approaches used to approximate the Bayes predictor with precision are detailed in Section 2.2.4.

2.2.1. Classification Problem

In classification, the learning algorithm (learner) is provided with a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$ of size $N \in \mathbb{N}$, where $\mathcal{X} = \mathbb{R}^d$ is the input (instance) space and $\mathcal{Y} = \{0, 1, \ldots, M-1\} = [M]_0$, $M \in \mathbb{N}$ the output (categorical class labels) space, and the $(\boldsymbol{x}_i, y_i)$ are assumed to be independent and identically distributed (i.i.d.) according to $p_{(X,Y)}(\cdot)$ [191]. According to the statistical learning theory, the primary goal of the learner in standard classification is to induce a hypothesis $h\colon \mathcal{X} \to \mathcal{Y}$, $h \in \mathcal{H}$, with low generalization error (risk):

$$R(h) = \mathbb{E}[\ell(y, h(\boldsymbol{x}))] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h(\boldsymbol{x}))\, d\, p_{(X,Y)}(\boldsymbol{x}, y)\,, \tag{2.4}$$

where $\mathcal{H}$ is the underlying hypothesis space (set of candidate functions the learner can choose from), $\ell\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ a loss function, and $p_{(X,Y)}(\cdot)$ is the joint probability measure modeling the underlying data-generating process. A loss function commonly used in standard classification is the 0-1 loss $\ell_{01}(y, \hat{y}) := [\![h(\boldsymbol{x}) \neq y]\!]$, where $[\![\cdot]\!]$ is the indicator function as defined in Section 1.3. The risk minimizer h^*, if it exists, achieves the minimum expected loss $\mathbb{E}[\ell(\cdot)]$ in terms of the loss function ℓ across the entire joint distribution $p_{(X,Y)}(\cdot)$ and is defined as $h^* = \arg\min_{h\in\mathcal{H}} R(h)$. If no such h^* exists, arg min should be interpreted as achieving the *infimum* over the hypothesis space $\mathcal{H}$.

The measure $p_{(X,Y)}(\cdot)$ in eq. (2.4) induces marginal probability (density or mass) functions on $\mathcal{X}$ and $\mathcal{Y}$ as well as a conditional probability of the class Y given an input instance $\boldsymbol{x}$, i.e., $p_{(X,Y)}(\boldsymbol{x}, y) = p_{Y|X}(y \,|\, \boldsymbol{x}) \times p_X(\boldsymbol{x})$. In practice, the learner does not observe these probabilities, so directly minimizing the risk eq. (2.4) is not feasible. Instead, learning in a standard classification setting

is commonly accomplished by minimizing (a regularized version of) *empirical risk* for h:

$$R_{\text{emp}}(h) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, h(\boldsymbol{x}_i)) \,. \tag{2.5}$$

In the subsequent discussions, the (learned) hypothesis that minimizes eq. (2.5), i.e., the empirical risk minimizer, is denoted by $g = \arg\min_{h \in \mathcal{H}} R_{\text{emp}}(h)$ [191]. In practice, for the available finite sampled data set $\mathcal{D}$, g is the best possible approximation (empirical estimation) of the true risk minimizer h^* and is an appropriation thereof.

Bayes-optimal Predictor

In statistical learning theory, the *Bayes predictor* is the optimal (pointwise) classification function $g^b \colon \mathcal{X} \to \mathcal{Y}$ producing a minimum expected risk, i.e., minimizing eq. (2.4) for a given loss function $\ell(\cdot)$:

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

where $\mathbb{E}_y[\ell]$ is the expected loss of the prediction $\hat{y}$ with respect to $y \in \mathcal{Y}$, and $p_{Y|X}(y \,|\, \boldsymbol{x})$ is the conditional probability of the class y given an input instance $\boldsymbol{x}$ [47]. The Bayes predictor simplifies in case of 0-1 the loss as

$$g^{bc}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} p_{Y|X}(y \,|\, \boldsymbol{x}) \,. \tag{2.6}$$

The expected loss of this predictor is known as the *Bayes error rate* $\mathrm{m}_{\text{ER}}\,(g^{bc})$.To determine the Bayes predictor, the conditional probability distribution $p_{Y|X}(\cdot)$ must be known for the underlying data set $\mathcal{D}$.

In the case where $\mathcal{X}$ and $\mathcal{Y}$ are independent of each other, i.e., $p_{Y|X}(y \,|\, \boldsymbol{x})$ corresponds to the marginal distribution $p_Y(y)$ as input features are completely uninformative. Consequently, the *marginal Bayes predictor* minimizing 0-1 loss

using only $\mathcal{Y}$ is defined as

$$g^{bc}(\boldsymbol{x}) \equiv g^{mc}(\boldsymbol{x}) \equiv \arg\max_{y \in \mathcal{Y}} p_Y(y)\,. \tag{2.7}$$

It assigns to each input $\boldsymbol{x}$ a class label from the set of labels with the highest marginal probability. Strictly speaking, as the maximum in eq. (2.7) is not necessarily unique, the marginal Bayes predictor may pick any label with the highest probability — however, it is assumed that the same label is selected for every $\boldsymbol{x}$, making it a constant function.

Probabilistic Classification

Distinct from standard classifiers, probabilistic classifiers focus on estimating the (conditional) class probabilities $p_{Y|X}(y \,|\, \boldsymbol{x})$ for each class y in $\mathcal{Y}$, given an input instance $\boldsymbol{x} \in \mathcal{X}$, with the predictions of that kind are denoted by $\hat{p}_{Y|X}(y \,|\, \boldsymbol{x})$. 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 $h_p \colon \mathcal{X} \to \mathbb{P}(\mathcal{Y})$ 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 $\ell_p \colon \mathcal{Y} \times \mathbb{P}(\mathcal{Y}) \to \overline{\mathbb{R}}$, where $\mathbb{P}(\mathcal{Y})$ denotes the set of PMFs on $\mathcal{Y}$.

Since $\mathcal{Y}$ is finite and consists of M classes, $\mathbb{P}(\mathcal{Y})$ can be represented by the $(M-1)$-simplex, i.e., predictions are denoted by the probability vector $h_p(\boldsymbol{x}) = \hat{\boldsymbol{p}} = (\hat{p}_0, \ldots, \hat{p}_{M-1})$, where $\hat{p}_m = \hat{\boldsymbol{p}}[m] = \hat{p}_{Y|X}(m \,|\, \boldsymbol{x})$ is the probability assigned to class m. Typically, learning predictors of that kind involve minimizing the *empirical risk*

$$R_{\mathrm{emp}|p}(h_p) = \frac{1}{N} \sum_{i=1}^{N} \ell_p(y_i, h_p(\boldsymbol{x}_i))\,. \tag{2.8}$$

Subsequently, in this thesis, the empirical risk minimizer is denoted by $g_p = \arg\min_{h_p \in \mathcal{H}_p} R_{\text{emp}|p}(h_p)$ [19, chap. 4][191]. In the case where Y is independent of X, the marginal Bayes predictor or estimated marginal Bayes predictor (marginal classifier denoted by g_p^{mc} in eq. (3.3)) is again the best constant probability predictor, i.e., the one with the lowest risk eq. (2.8) among all constant predictors.

Deterministic Prediction A probabilistic prediction may also serve as a basis for a deterministic prediction if needed. Typically, the class label with the highest (predicted) probability is adopted in that case, i.e., $\hat{y} = \arg\max_{y \in \mathcal{Y}} \hat{p}_{Y|X}(y \mid \boldsymbol{x})$. As readily seen, this prediction minimizes the standard 0-1 loss in expectation. The other way around, this also shows that to perform well in terms of standard 0-1 loss, the learner merely needs to identify the most probable class, or, stated differently, strong performance can even be achieved with relatively inaccurate estimates, provided the highest predicted probability is still assigned to the truly most probable class label [48, 30].

2.2.2. Learning and Evaluation Measures

This section describes the loss functions used for learning (probabilistic) classifiers and evaluation metrics used to evaluate the performance of the learned classifiers.

Loss Functions

This section defines an essential class of loss functions used for learning a probabilistic classifier, which is so-called (strictly) *proper scoring rules* [73]. Roughly speaking, such losses incentivize the learner to predict accurate probabilities. Formally, a loss $\ell_p(\cdot)$ is a proper scoring rule if

$$\mathbb{E}_{y \sim p_{Y|X}(\cdot \mid \boldsymbol{x})}[\ell_p(y, p_{Y|X}(y \mid \boldsymbol{x}))] \leq \mathbb{E}_{y \sim p_{Y|X}(\cdot \mid \boldsymbol{x})}[\ell_p(y, \hat{p}_{Y|X}(y \mid \boldsymbol{x}))]\,,$$

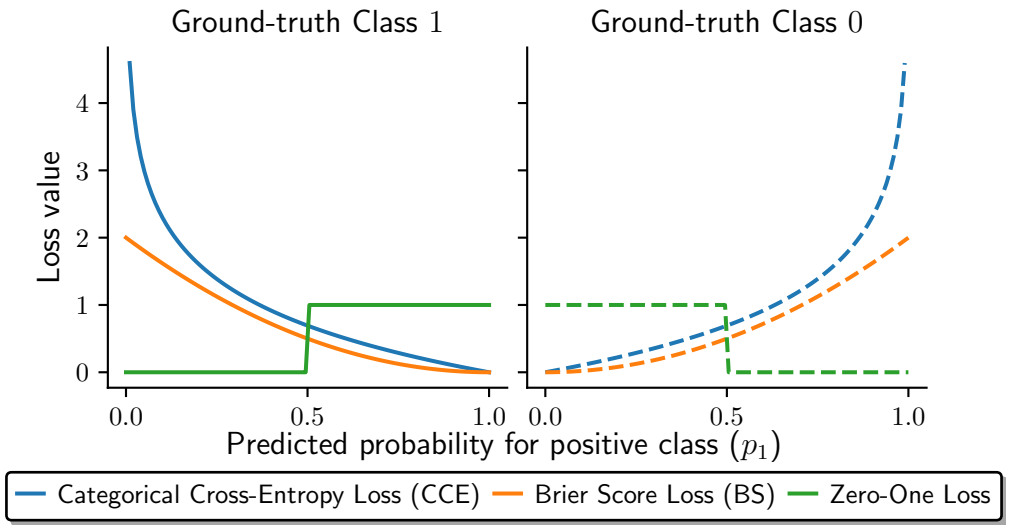


Figure 2.2.: Behavior of various proper loss functions in binary classification setting

for all distributions $p_{Y|X}(\cdot), \hat{p}_{Y|X}(\cdot)$, and a *strictly* proper scoring rule is the above inequality is strict whenever $\hat{p}_{Y|X}(\cdot) \neq p_{Y|X}(\cdot)$. Thus, by predicting the true distribution $p_{Y|X}(\cdot)$ of the (categorical) random variable Y, the learner minimizes its loss in expectation, making them apt for learning a probabilistic classifier. Important examples of strictly proper scoring rules include the Brier score (BS) and the categorical cross-entropy (CCE), as shown in Figure 2.2. In this work, the CCE is used for training the multi-layer perceptron (MLP) models in the AutoGluon tool for the experiments in Chapters 4 and 5 and convolutional neural networks (CNNs) optimized using neural architecture search (NAS) for the experiments in Chapter 6.

Categorical Cross-entropy The CCE is defined as

$$\ell_{\text{CCE}}(y, \hat{\boldsymbol{p}}) := -\ln(\hat{p}_y) = -\ln(\hat{\boldsymbol{p}}[y])\,.$$

The CCE loss (logarithmic scoring rule) [19, chap. 4] is widely recognized for its information-theoretic interpretations and practical effectiveness [75, 173, 44]. It can also be motivated in the realm of Neyman–Pearson theory [59]. This

motivation for choosing CCE over other common loss functions stems from its ability to ensure reliable and generalizable performance across experiments when learning MLPs and CNNs.

Brier score The BS is defined as

$$\ell_{\mathrm{BS}}(y, \hat{\boldsymbol{p}}) := -\sum_{m=0}^{M-1} (\llbracket y = m \rrbracket - \hat{p}_m)^2 = -\sum_{m=0}^{M-1} (\llbracket y = m \rrbracket - \hat{\boldsymbol{p}}[m])^2$$

The BS loss (quadratic scoring rule) has axiomatic characteristic [73, 22]

Zero-one Score The zero-one loss function measures whether the predicted class (the one with the highest predicted probability) matches the true class, which is formally defined as

$$\ell_{p01}(y, \hat{\boldsymbol{p}}) := \llbracket y = \arg\max_{m \in [M]_0} \hat{p}_m \rrbracket = \llbracket y = \arg\max_{m \in [M]_0} \hat{\boldsymbol{p}}[m] \rrbracket \,. \tag{2.9}$$

The zero-one score is simple and interpretable, measuring accuracy by assigning a 1 score if the predicted class matches the true class and 0 otherwise.It rewards predictions based on the mode of the probability distribution, with reduced scores for multiple modes, but does not define a Bregman divergence due to its non-differentiable and non-convex entropy function [73].

Metrics

Koyejo et al., Powers (2015, 2011) [110, 154] define 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 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and the predictions denoted by the vector $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_N)$. The predictions could be obtained by the empirical risk minimizer g derived using eq. (2.5) for a standard clas-

sification, such that $\hat{y}_i = g(\boldsymbol{x_i}), \forall i \in [N]$ or by the deterministic predictions produced by a probabilistic classifier g_p derived using eq. (2.8), such that $\hat{y}_i = \arg\max_{m \in M} g_p(\boldsymbol{x_i})[m], \forall i \in [N]$.

Confusion matrix (CM) Many evaluation metrics are based on the concepts *true positive* (m_{TP}), *true negative* (m_{TN}), *false positive* (m_{FP}), and *false negative* (m_{FN}), ground-truth positives (N_p), ground-truth negatives (N_n), predicted positives ($\hat{N}_p$) and predicted negatives ($\hat{N}_n$,) which are defined as

$$\mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N} [\![y_i = 0, \hat{y}_i = 0]\!], \; \mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N} [\![y_i = 1, \hat{y}_i = 1]\!],$$
$$\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N} [\![y_i = 0, \hat{y}_i = 1]\!], \; \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{N} [\![y_i = 1, \hat{y}_i = 0]\!],$$
$$N_p = \sum_{i=1}^{N} [\![y_i = 1]\!], \; N_n = \sum_{i=1}^{N} [\![y_i = 0]\!], \; \hat{N}_p = \sum_{i=1}^{N} [\![\hat{y}_i = 1]\!], \; \hat{N}_n = \sum_{i=1}^{N} [\![\hat{y}_i = 0]\!] .$$

The confusion matrix (CM) is defined using these metrics as

$$\mathrm{m}_{\mathrm{CM}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \begin{pmatrix} \mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) & \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \\ \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) & \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \end{pmatrix} . \tag{2.10}$$

Table 2.1 present the common evaluation metrics and how they can be calculated using the CM entities.

Accuracy It is defined as the proportion of correct predictions

$$\begin{aligned} \mathrm{m}_{\mathrm{ACC}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) &:= \frac{1}{N} \sum_{i=1}^{N} [\![y_i = \hat{y}_i]\!] = \frac{\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{N} \\ &= \frac{\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})} . \end{aligned}$$

Error rate It is defined as the proportion of incorrect predictions

$$\begin{aligned}\mathrm{m}_{\mathrm{ER}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) &:= \frac{1}{N}\sum_{i=1}^{N}[\![y_i \neq \hat{y}_i]\!] = \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{N} \\ &= \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}\,.\end{aligned}$$

False negative rate (FNR) It is defined as the ratio of *false negatives* to the number of positive instances.

$$\mathrm{m}_{\mathrm{FNR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 1 - \mathrm{m}_{\mathrm{TPR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}\,.$$

False positive rate (FPR) It is defined as the ratio of *false positives* to the number of negative instances

$$\mathrm{m}_{\mathrm{FPR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 1 - \mathrm{m}_{\mathrm{TNR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}\,.$$

N	$\hat{N}_p$	$\hat{N}_n$	**Metrics**
N_p	m_{TP}	m_{FN}	$\mathrm{m}_{\mathrm{TPR}} = \frac{\mathrm{m}_{\mathrm{TP}}}{N_p}$, $\mathrm{m}_{\mathrm{FNR}} = \frac{\mathrm{m}_{\mathrm{FN}}}{N_p}$
N_n	m_{FP}	m_{TN}	$\mathrm{m}_{\mathrm{FPR}} = \frac{\mathrm{m}_{\mathrm{FP}}}{N_n}$, $\mathrm{m}_{\mathrm{TNR}} = \frac{\mathrm{m}_{\mathrm{TN}}}{N_n}$
$\mathrm{m}_{\mathrm{PRE}} = \frac{N_p}{N_p+N_n}$	$\mathrm{m}_{\mathrm{PPV}} = \frac{\mathrm{m}_{\mathrm{TP}}}{\hat{N}_p}$	$\mathrm{m}_{\mathrm{FOR}} = \frac{\mathrm{m}_{\mathrm{FN}}}{\hat{N}_n}$	$\mathrm{m}_{\mathrm{BER}} = \frac{\mathrm{m}_{\mathrm{FNR}}+\mathrm{m}_{\mathrm{FPR}}}{2}$
$\mathrm{m}_{\mathrm{ACC}} = \frac{\mathrm{m}_{\mathrm{TP}}+\mathrm{m}_{\mathrm{TN}}}{N}$	$\mathrm{m}_{\mathrm{FDR}} = \frac{\mathrm{m}_{\mathrm{FP}}}{\hat{N}_p}$	$\mathrm{m}_{\mathrm{NPV}} = \frac{\mathrm{m}_{\mathrm{TN}}}{\hat{N}_n}$	
$\mathrm{m}_{\mathrm{F1}} = \frac{2\,\mathrm{m}_{\mathrm{PPV}}\,\mathrm{m}_{\mathrm{TPR}}}{\mathrm{m}_{\mathrm{PPV}}+\mathrm{m}_{\mathrm{TPR}}} = \frac{2\,\mathrm{m}_{\mathrm{TP}}}{2\,\mathrm{m}_{\mathrm{TP}}+\mathrm{m}_{\mathrm{FP}}+\mathrm{m}_{\mathrm{FN}}}$			
$\mathrm{m}_{\mathrm{MCC}} = \sqrt{\mathrm{m}_{\mathrm{TPR}}\,\mathrm{m}_{\mathrm{TNR}}\,\mathrm{m}_{\mathrm{PPV}}\,\mathrm{m}_{\mathrm{NPV}}} - \sqrt{\mathrm{m}_{\mathrm{FNR}}\,\mathrm{m}_{\mathrm{FPR}}\,\mathrm{m}_{\mathrm{FOR}}\,\mathrm{m}_{\mathrm{FDR}}}$			

Table 2.1.: Classification evaluation metrics

Mathews correlation coefficient (MCC) It is a balanced measure that considers true and false positives, including true and false negatives.It produces a value between -1 and $+1$, where $+1$ represents a perfect prediction, 0 is

random, and -1 indicates total disagreement between the predictions and ground truths. It is formally defined as

$$\mathrm{m}_{\mathrm{MCC}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \left(\frac{(\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \times \mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) - \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \times \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}))}{\sqrt{(\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}))(\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}))}} \right) \times \left(\frac{1}{\sqrt{(\mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}))(\mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}))}} \right) .$$

Balanced error-rate (BER) It is the average of false positive rate (FPR) and false negative rate (FNR), offering a balanced evaluation metric for a given classification model as

$$\begin{aligned} \mathrm{m}_{\mathrm{BER}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= \frac{1}{2} \left(\frac{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{TN}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FP}}(\boldsymbol{y}, \hat{\boldsymbol{y}})} + \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\mathrm{m}_{\mathrm{TP}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FN}}(\boldsymbol{y}, \hat{\boldsymbol{y}})} \right) \\ &= \frac{\mathrm{m}_{\mathrm{FNR}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \mathrm{m}_{\mathrm{FPR}}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{2} . \end{aligned}$$

2.2.3. Classifier Calibration

Ensuring the precision of probability estimates from the probabilistic risk minimizer g_p is crucial for the effectiveness of the Log-Loss and the probabilitiy-corrected softmax (PC-softmax) baseline MI estimation approach, described in Section 3.3. However, even with the CCE loss function, these probabilities might not accurately reflect the ground-truth conditionals $p_{Y|X}$, especially in imbalanced class scenarios [140, 48]. Famous classification methods like Naive Bayes, random forest classifier (RF) produce inaccurate probability estimates despite their success in accurate classification [30, 48]. Consequently, using calibration techniques becomes essential to produce predicted probabilities close to actual frequencies, thereby enhancing the predictive value of probabilistic models [63]. For calibration, the post-processing methods map predicted class probabilities to more reliable and calibrated probabilities, often achieved through distinct models or transformation methods [63]. This work focuses

on five multi-class calibration methods to improve MI estimation using the Log-Loss approach [81]: Platt's Scaling, Isotonic Regression, Beta Calibration, Temperature Scaling, and Histogram Binning Section 3.3.2.

Platt's Scaling

Platt's scaling is a logistic regression-based calibration method designed for binary classification, adjusting raw scores (logits) from a classifier to produce calibrated probabilities [151, 152]. In the binary classification setting $M = 2$, it transforms the real-valued score $s_1 \in \mathbb{R}$ of the positive class from the classifier into a calibrated probability using a logistic (sigmoid) function $f_1(s_1; A, B) \colon \mathbb{R}^3 \to (0, 1)$ is defined as

$$f_1(s_1; A, B) = \frac{1}{1 + \exp(-A \cdot s_1 - B)} ,$$

where $A \in \mathbb{R}$ is the weight parameter controlling the slope, and $B \in \mathbb{R}$ is the bias parameter shifting the curve.

These parameters are learned by maximizing the log-likelihood of the observed data using the binary cross-entropy (BCE) loss on the given training binary classification dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is defined as

$$\ell_{\text{BCE}}(\boldsymbol{s}; A, B) = -\frac{1}{N} \sum_{i=1}^{N} y_i \ln(f_1(s_{i1}; A, B)) + (1 - y_i) \ln(1 - f_1(s_{i1}; A, B)) ,$$

where $y_i \in \{0, 1\}$ denotes the true label for instance $i \in [N]$.

Multi-class Classification Platt's scaling extends to multi-class classification using the one-versus-rest (OVR) approach [77], where each class $m \in [M]_0$ is independently calibrated by treating it as positive and all others as negative. The corresponding binary dataset for class m is $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ using the

following label transformation

$$\tilde{y}_i = \begin{cases} 1, & \text{if } y_i = m \\ 0, & \text{elif } y_i \neq m \end{cases}$$

It is applied independently for each class m, resulting in M separate logistic regression models with their parameters A_m and B_m, using a logistic (sigmoid) function $f_m(s_m; A_m, B_m)\colon \mathbb{R}^3 \to (0,1)$ is defined as

$$f_m(s_m; A_m, B_m) = \frac{1}{1 + \exp(-A_m \cdot s_m - B_m)}\,,$$

where s_m is the raw score for class m. The parameters A_m and B_m are estimated by optimizing the BCE for the binary classification dataset $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ using $\ell_{\text{BCE}}(\boldsymbol{s}; A, B)$.

The likelihood is represented by the sum of log-likelihoods for each independent function for class m to evaluate the CCE over the given classification dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is defined as

$$\ell_{\text{CCE}}(\boldsymbol{s}; \boldsymbol{A}, \boldsymbol{B}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{m=0}^{M-1} [\![y_i = m]\!] \log(f_m(s_{im}; A_m, B_m))\,,$$

where $\boldsymbol{A} = (A_0, A_1, \ldots, A_{M-1})$, $\boldsymbol{B} = (B_0, B_1, \ldots, B_{M-1})$ and $[\![y_i = m]\!]$ ensures that only the log-probability of the true class is considered for each sample.

The calibrated probabilities are normalized to sum up to 1 as

$$\tilde{p}_{im} = \frac{f_m(s_{im}; A_m, B_m)}{\sum_{j=0}^{M-1} f_j(s_{ij}; A_j, B_j)}$$

It is easy to implement and efficient to train using standard logistic regression with a convex loss. However, it has limitations, such as the fixed sigmoid shape, which can lead to over-confidence and overestimation.The logistic regression model assumes that the inputs are real-valued scalars, suitable for SVM

margins but not for probabilistic classifiers where inputs are in the interval $[0, 1]$. Therefore, transformations like logits are necessary when calibrating probabilistic classifiers [63].

Beta Calibration

Beta calibration is a method that extends the flexibility of Platt's Scaling by utilizing Beta distributions, which allow a more adaptable mapping between predicted scores and calibrated probabilities [112]. This approach can capture a broader range of shapes than the logistic function used in Platt's Scaling.

Binary Classification In the binary classification setting, it aims to transform the un-calibrated predicted probability $\hat{p}_1$ of the positive class from the classifier into a calibrated probability that more accurately reflects the true likelihood of the positive class using a Beta distribution, using a function $f_1(\hat{p}_1; a, b, c) \colon [0, 1] \times \mathbb{R}^3 \to (0, 1)$ is defined as

$$f_1(\hat{p}_1; a, b, c) = \frac{1}{1 + \exp(-(a \ln(\hat{p}_1) + b \ln(1 - \hat{p}_1) + c))} ,$$

where the parameters $a \in \mathbb{R}$ and $b \in \mathbb{R}$ controls the influence of $\ln(\hat{p}_1)$ and $\log(1 - \hat{p}_1)$, respectively and $c \in \mathbb{R}$ is a location parameter (bias term).

These parameters are learned by minimizing the negative log-likelihood of the calibrated probabilities using the log-loss or BCE for the given training binary classification dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is defined as

$$\ell_{\text{BCE}}(\hat{\boldsymbol{p}}; a, b, c) = -\frac{1}{N} \sum_{i=1}^{N} y_i \ln(f_1(\hat{p}_{i1}; a, b, c)) + (1 - y_i) \ln(1 - f_1(\hat{p}_{i1}; a, b, c)) ,$$

where $y_i \in \{0, 1\}$ is the true label, and $f_1(\hat{p}_{i1}; a, b, c)$ represents the calibrated probability for the positive class with predicted probability $\hat{p}_{i1}$, for $i \in [N]$ instance.

Multi-class Classification Beta calibration extends to multi-class classification using the OVR approach [112, 113], where each class $m \in [M]_0$ is independently calibrated against all others. The corresponding binary dataset for class m is $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ using the following label transformation

$$\tilde{y}_i = \begin{cases} 1, & \text{if } y_i = m \\ 0, & \text{elif } y_i \neq m \end{cases}$$

The Beta distribution function $f_m(\hat{p}_m; a_m, b_m, c_m) \colon [0,1] \times \mathbb{R}^3 \to (0,1)$ is applied independently for each class m, resulting in M separate functions with their own set of parameters a_m, b_m and c_m is defined as

$$f_m(\hat{p}_m; a_m, b_m, c_m) = \frac{1}{1 + \exp(-(a_m \log(\hat{p}_m) + b_m \log(1 - \hat{p}_m) + c_m))}$$

where s_m is the predicted probability of class m. The parameters A_m and B_m are estimated by minimizing the BCE for the binary classification dataset $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ using $\ell_{\text{BCE}}(\hat{\boldsymbol{p}}; a_m, b_m, c_m)$.

The likelihood is represented by the sum of log-likelihoods for each independent function for class m to evaluate the CCE over the given classification dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is defined as

$$\ell_{\text{CCE}}(\hat{\boldsymbol{p}}; \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = -\frac{1}{N} \sum_{i=1}^{N} \log \left(\sum_{m=0}^{M-1} [\![y_i = m]\!] \cdot f_m(\hat{p}_{im}; a_m, b_m, c_m) \right)$$

where $\boldsymbol{a} = (a_0, a_1, \dots a_{M-1})$, $\boldsymbol{b} = (b_0, b_1, \dots b_{M-1})$, and $\boldsymbol{c} = (c_0, c_1, \dots c_{M-1})$ and $[\![y_i = m]\!]$ ensures that only the log of predicted probability $\hat{p}_{im}$ of the true class m is considered for each i^{th} sample.

The calibrated probabilities are normalized to sum up to 1 as

$$\tilde{p}_{im} = \frac{f_m(\hat{p}_{im}; a_m, b_m, c_m)}{\sum_{j=0}^{M-1} f_j(\hat{p}_{ij}; a_j, b_j, c_j)}$$

An alternative to the OVR approach is calibration using the multinomial logistic regression, which learns unified parameters a_m, b_m, and c_m across classes $m \in [M]_0$, directly adjusting the vector of predicted probabilities to account for inter-class relationships. It allows flexibility with diverse maps, such as inverse sigmoids and identity functions, making it adaptable for two-class probabilistic classifiers. However, its limitation to specific calibration functions may not suit classifiers requiring complex adjustments, where non-parametric methods could be more effective [63].

Temperature Scaling

Temperature scaling is a calibration method that scales the logits (log-odds) of predicted probabilities using a single "temperature" parameter $t \in \mathbb{R}$ and applies a linear operation before the softmax. This method calibrates the entire distribution while maintaining the rank order of class labels according to originally predicted class probabilities [77].

It calibrates the predicted probabilities $\hat{p}_m$ for each class m using the logits vector $\boldsymbol{z} = [z_0, z_1, \ldots, z_{M-1}]$ and a temperature parameter $t \in \mathbb{R}^+$ using the (softmax) function $c_m(\boldsymbol{z}; t) \colon \mathbb{R}^M \times \mathbb{R}^+ \to (0,1)^M$ defined as

$$c_m(\boldsymbol{z}; t) = \frac{\exp\left(\frac{z_m}{t}\right)}{\sum_{j=0}^{M-1} \exp\left(\frac{z_j}{t}\right)}$$

If logits $\boldsymbol{z}$ are not available, logit transformation can be applied to predicted probabilities as

$$z_m = \operatorname{logit}_M(\hat{p}_m) = \ln \frac{\hat{p}_m}{\hat{p}_{M-1}} \,,$$

where $M-1$ is the reference class to compute the probability ratio with the remaining classes. The parameter is also normally estimated with the log-loss, and the objective function to evaluate the CCE over the given classification

dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ as

$$\ell_{\mathrm{CCE}}(\boldsymbol{z}; t) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{m=1}^{M} [\![y_i = m]\!] \ln \left(c_m(\mathbf{z}_i; t) \right)$$

It is a simple calibration method that adjusts logits using a single temperature parameter t. This approach is efficient, easy to implement, and handles multi-class calibration without using OVR. The single parameter helps prevent over-fitting in small datasets but can be limiting if the required calibration map is not within its function space. While effective for reducing overconfidence in deep learning (DL) models, its performance is sub-optimal for more complex calibration requirements [63].

Isotonic Regression

Isotonic regression is used in probability calibration to fit a non-decreasing, piecewise constant function that maps predicted probabilities to calibrated probabilities. This technique is particularly effective when the predicted probabilities do not accurately reflect the true likelihoods of events, often seen in machine learning models like decision trees or random forests [7, 205].

Binary Classification In the case of binary output variable $M = 2$, it adjusts predicted probabilities so that they form a non-decreasing sequence or have a monotonic reliability diagram. For the positive class $m = 1$, it learns a binning function f_m with parameters $\boldsymbol{b} = \{0, b_1, b_2, \ldots, b_B\}, \forall j \in [B], b_j \in [0, 1], b_j < b_{j+1}$ and $\boldsymbol{v} = \{v_1, v_2, \ldots, v_B\}, \forall j \in [B], b_j \in [0, 1], v_j \leq v_{j+1}$, where B is the number of bins. The calibrated function $f_1 \colon [0, 1]^3 \to [0, 1]$ for the positive class $m = 1$ with predicted probability p_1 is defined as:

$$f_1(\hat{p}_1; \boldsymbol{b}, \boldsymbol{v}) = \sum_{j=1}^{B-1} [\![\hat{p}_1 \geq b_j]\!] \cdot [\![\hat{p}_1 < b_{j+1}]\!] \cdot v_j \,,$$

where $\hat{p}_1$ is the predicted probability of the positive class, and v_j is the calibrated value learned from the data for bin $j \in [B]$, reflecting its average calibrated probability.

The fitted function f_1 is determined by minimizing the mean squared error (MSE) or BS between calibrated probabilities and true labels for the given training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ defined as

$$\ell_{\text{BS}}(\hat{\boldsymbol{p}}; \boldsymbol{b}, \boldsymbol{v}) = \frac{1}{N} \sum_{i=1}^{N} \left(f_1(\hat{p}_{i1}; \boldsymbol{b}, \boldsymbol{v}) - y_i\right)^2$$

where $y_i \in \{0, 1\}$ is the true label, and $\hat{p}_{i1}$ is the predicted probability of the positive class for data point $i \in [N]$. The parameters are learned using the pool adjacent violators algorithm (PAVA) to ensure the monotonicity of $\boldsymbol{v}$. If v_j does not maintain monotonicity (i.e., $v_j > v_{j+1}$), adjacent bins are merged, and v_j is recalculated as the mean true outcome of the merged bins [205].

Multi-class Classification Isotonic regression extends via a OVR approach for multi-class classification with M classes, where each class is independently calibrated against all others [205]. Specifically, for each class $m \in [M]_0$, the class m is treated as positive, and the remaining classes as negative. The corresponding binary dataset for class m is $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ using the following label transformation

$$\tilde{y}_i = \begin{cases} 1, & \text{if } y_i = m \\ 0, & \text{otherwise} \end{cases}$$

A binning (calibration) function $f_m \colon [0,1]^3 \to [0,1]$ with parameters $\boldsymbol{b}_m = \{0, b_{m1}, b_{m2}, \ldots, b_{mB_m}\}$ is learned for class m, where $b_{mj} \in [0,1]$ and $b_{mj} < b_{m(j+1)}$ and $\boldsymbol{v}_m = \{v_{m1}, v_{m2}, \ldots, v_{mB_m}\}$ where $v_{mj} \in [0,1]$ and $v_{mj} \leq v_{m(j+1)}$. The function f_m for class m, learned by minimizing $\ell_{\text{BS}}(\cdot)$ on $\mathcal{D}_m$, is defined

as

$$f_m(\hat{p}_{im}; \boldsymbol{b}_m, \boldsymbol{v}_m) = \sum_{j=1}^{B_m-1} [\![\hat{p}_{im} \geq b_{mj}]\!] \cdot [\![\hat{p}_{im} < b_{m(j+1)}]\!] \cdot v_{mj}$$

$$\ell_{\text{BS}}(\hat{\boldsymbol{p}}; \boldsymbol{b}_m, \boldsymbol{v}_m) = \frac{1}{N} \sum_{i=1}^{N} \left(f_m(\hat{p}_{im}; \boldsymbol{b}_m, \boldsymbol{v}_m) - \tilde{y}_i\right)^2 ,$$

where $\tilde{y}_i = 1$ if the true class of instance i is m, otherwise 0.

The calibrated probabilities are normalized to sum up to 1 as

$$\tilde{p}_{im} = \frac{f_m(\hat{p}_{im}; \boldsymbol{b}_m, \boldsymbol{v}_m)}{\sum_{j=0}^{M-1} f_m(\hat{p}_{ij}; \boldsymbol{b}_j, \boldsymbol{v}_j)}$$

It is a robust method for probability calibration, which can identify optimal bin edges that minimize the BS under the monotonicity assumption, effectively capturing the convex hull in Receiver Operating Characteristic (ROC) analysis. Its non-parametric nature prevents significant model misfits, offering flexibility in various scenarios. In multi-class settings, it independently calibrates each class using a OVR approach, producing class-specific calibration functions with distinct parameters tailored to the predicted score distribution of each class [63].

Despite its strengths, isotonic regression has limitations.It tends to predict constant values within bins, which can increase grouping loss and may yield sub-optimal results when the un-calibrated model's reliability diagram is non-monotonic. Additionally, it can be resource-intensive in terms of training time and memory on large datasets and often results in extreme confidence values (0 and 1) in the outer bins, necessitating output clipping to avoid issues with evaluation metrics like log-loss [63].

Histogram Binning

Histogram binning is a non-parametric calibration method that provides better-calibrated probabilities to reflect true event likelihoods [204]. Similar to Isotonic regression, it involves partitioning the probability space into bins and assigning calibrated values based on observed frequencies within these bins.Unlike Isotonic regression, Histogram binning does not enforce monotonicity and discretizes the probability space, making it suitable for cases where a non-monotonic mapping may be necessary [204].

Binary Classification A binary output variable $M = 2$ adjusts predicted probabilities by assigning them to discrete bins associated with a calibrated value. For the positive class $m = 1$, the method defines a binning function f_1 with bin edges $\boldsymbol{b} = \{0, b_1, \ldots, b_B\}$ and calibrated probabilities $\boldsymbol{v} = \{v_1, v_2, \ldots, v_B\}$. Each bin boundary b_j is calculated as $b_j = \frac{j}{B}, \forall j \in [B] = \{1, \ldots, B\}$, such that $b_j \in [0, 1]$ and $b_j < b_{j+1}$.

Similar to Isotonic Regression, the calibrated function $f_1 \colon [0, 1]^3 \to [0, 1]$ for the positive class $m = 1$ with predicted probability p_1 is defined as

$$f_1(\hat{p}_1; \boldsymbol{b}, \boldsymbol{v}) = \sum_{j=1}^{B-1} [\![\hat{p}_1 \geq b_j]\!] \cdot [\![\hat{p}_1 < b_{j+1}]\!] \cdot v_j \,,$$

where $\hat{p}_1$ is the predicted probability of the positive class, and v_j is the calibrated value for bin $j \in [B]$.

The values $\boldsymbol{v}$ are determined by computing the mean observed frequency of the positive class within each bin using the training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ as

$$v_j = \sum_{i=1}^{N} \frac{[\![\hat{p}_{i1} \geq b_j]\!] \cdot [\![\hat{p}_{i1} < b_{j+1}]\!] \cdot [\![y_i = 1]\!]}{[\![\hat{p}_{i1} \geq b_j]\!] \cdot [\![\hat{p}_{i1} < b_{j+1}]\!]} \,,$$

where $[\![(y_i = 1)]\!]$ is equal to 1 if $y_i = 1$ and 0 otherwise, ensuring that v_j reflects the proportion of positive class occurrences within j^{th} bin, and v_j is the final calibrated probability for un-calibration value $\hat{p}_1$ lying in bin $j \in [B]$.

Multi-class Classification Histogram Binning extends via a OVR approach for multi-class classification with M classes, where each class is independently calibrated against all others [204]. Specifically, for each class $m \in [M]_0$, the class m is treated as positive, and all other classes are negative and create the corresponding binary classification dataset $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$ by transforming the labels as

$$\tilde{y}_i = \begin{cases} 1, & \text{if } y_i = m \\ 0, & \text{otherwise} \end{cases}$$

For each class m, a binning function f_m is learned with parameters $\boldsymbol{b}_m = \{0, b_{M1}, \ldots, b_{mB}\}$ and calibrated values $\boldsymbol{v}_m = \{v_{M1}, v_{m2}, \ldots, v_{mB}\}$, where each bin boundary b_{mj} is computed as $b_{mj} = \frac{j}{B}, \forall j \in [B]$, ensuring $b_{mj} \in [0, 1]$ and $b_{mj} < b_{m(j+1)}$.

The values in the vector $\boldsymbol{v}_m$ are determined by computing the mean observed frequency of the positive class within each bin j using the training dataset $\mathcal{D}_m = \{\boldsymbol{x}_i, \tilde{y}_i\}_{i=1}^N$:

$$v_{mj} = \sum_{i=1}^{N} \frac{[\![\hat{p}_{im} \geq b_{mj}]\!] \cdot [\![\hat{p}_{im} < b_{m(j+1)}]\!] \cdot [\![\tilde{y}_i = 1]\!]}{[\![\hat{p}_{im} \geq b_{mj}]\!] \cdot [\![\hat{p}_{im} < b_{m(j+1)}]\!]},$$

where $[\![(\tilde{y}_i = 1)]\!]$ if the true class of instance i is m, otherwise 0, ensuring that v_j reflects the proportion of class m occurrences within j^{th} bin, and v_{mj} is the final calibrated probability for un-calibration value $\hat{p}_{im}$ lying in bin $j \in [B]$.

The calibrated function f_m for each class m using the $\boldsymbol{v}_m$ is defined as

$$f_m(\hat{p}_{im}; \boldsymbol{b}_m, \boldsymbol{v}_m) = \sum_{j=1}^{B-1} [\![\hat{p}_{im} \geq b_{mj}]\!] \cdot [\![\hat{p}_{im} < b_{m(j+1)}]\!] \cdot v_{mj}$$

where $\hat{p}_1$ is the predicted probability of the positive class, and v_j is the calibrated value for bin $j \in [B]$.

The calibrated probabilities are normalized to sum up to 1 as

$$\tilde{p}_{im} = \frac{f_m(\hat{p}_{im}; \boldsymbol{b}_m, \boldsymbol{v}_m)}{\sum_{j=0}^{M-1} f_m(\hat{p}_{ij}; \boldsymbol{b}_j, \boldsymbol{v}_j)}$$

Histogram Binning is a straightforward calibration method that improves the reliability of predicted probabilities by discretizing them into bins and aligning them with observed class frequencies. Unlike Isotonic Regression, it does not enforce monotonicity between bins, offering flexibility for non-monotonic probability mappings and faster training.

2.2.4. Automated Machine Learning

In the last decade, the field of AutoML has emerged as a response to the unmet demand for engineering machine learning (ML) applications experts. While the ultimate goal of AutoML is to automate the entire data science workflow, the most extensively studied challenge is the combined algorithm selection and hyperparameter optimization (CASH), which was first formally specified by [189]. Since the introduction of CASH, various AutoML systems have been developed [189, 60, 139, 133], demonstrating promising results in optimizing the selection of ML algorithms and their hyperparameters for specific tasks, which generally involve a dataset and a loss function. A comprehensive overview of AutoML methods is available in [212]. AutoML aims to automate various stages of the ML workflow, from data pre-processing to model deployment, with its crucial aspect being hyperparameter optimization (HPO), which involves fine-tuning the internal settings of ML algorithms to optimize model performance [3]. NAS, often misunderstood as a distinct problem, is a subset of HPO, with both focusing on finding an optimal configuration within a defined search space, though NAS automates explicitly the design of neural network architectures, as illustrated in Figure 2.3.

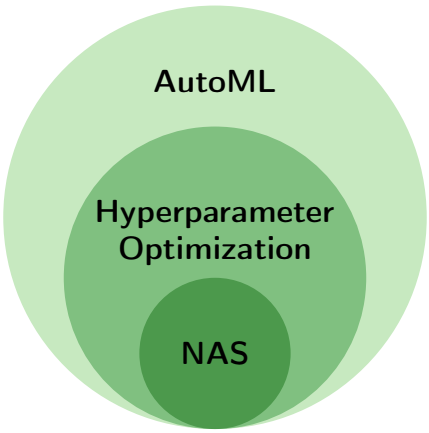


Figure 2.3.: Relation between AutoML and NAS [3]

Neural Architecture Search

NAS is a sub-field of AutoML that focuses specifically on optimizing neural network architectures for given learning tasks [3]. Unlike general AutoML approaches that aim to automate the entire ML pipeline, NAS is tailored exclusively for neural networks. It seeks to discover the best possible network architecture rather than selecting algorithms or tuning hyperparameters alone [3]. By systematically exploring various network designs, NAS automates the process of finding architectures that achieve high performance on a given task, ultimately optimizing the structure and efficiency of neural networks [3].

Search Strategies The search strategy in NAS significantly impacts search performance and runtime. Previous work applying NAS to SCA has primarily focused on Bayesian and Random search strategies [201], which are not always time-efficient [160, 117].This study examines four search strategies — Random, Greedy, Hyperband, and Bayesian — with a fixed budget of 1000 trials, utilizing their implementations provided by `AutoKeras` [100]. Typically, search strategies explore the search space by trying many substantially different architectures. This is followed by exploitation, in which well-performing architectures are further improved via small changes. Because each search algorithm

can only consider a limited number of architectures (1000 for experiments in Chapter 6), it should have a balanced trade-off between exploration and exploitation to perform efficiently and accurately.

Random The RANDOM search technique uniformly chooses a unique architecture configuration at random (with replacement) from the complete search space and evaluates its performance. This process is repeated for a specified number of trials but is preempted if it has exhausted the search space or the same configuration is chosen multiple times. This means it focuses only on exploration without any exploitation.

Greedy The GREEDY search algorithm proposed by Jin [99] works in two distinct stages, exploration and exploitation. In the exploration stage, it evaluates uniformly randomly chosen models for a limited number of trials [100]. In the exploitation stage, it generates models that are neighboring to the best-performing model from the first stage and exclusively attempts to improve it. This is done by traversing a hierarchical hyperparameter tree representing the hyperparameters of the best-performing model, as well as possible changes such that the new hyperparameter values remain close to that of the best-performing model with high probability [99]. This tree is rebuilt if a better architecture is found, at which point this architecture becomes the starting point for subsequent exploitation. The algorithm continues the exploitation process until either the trial limit is reached or until it has exhausted the entire search space. The GREEDY search algorithm is time-efficient, but it might get stuck in local optima since it performs limited exploration, e.g., for only 1 % of the maximum number of trials in `AutoKeras` [99, 100].

Hyperband The HYPERBAND search algorithm is based on the successive halving algorithm [117]. The Successive Halving algorithm equally divides the resources (time, epochs) into several hyperparameter configurations. Each time step (2-3 epochs) checks their performance and, at the end, keeps the top-half

best-performing configurations. This process is repeated until only the best-performing configurations are left. The HYPERBAND algorithm is time-efficient and balances the trade-off between exploration (check many models with a low budget) and exploitation (provide a high budget to the best-performing architectures) very well [117].

Bayesian The BAYESIAN search algorithm is based on Bayesian optimization, which assumes a (black-box) function $f' \colon \boldsymbol{\Theta} \to \mathbb{R}$ over the hyperparameter space $\boldsymbol{\theta} \in \boldsymbol{\Theta} = (\Theta_1, \ldots, \Theta_n)$, such that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ represents one hyperparameter configuration [61]. Each hyperparameter space can be an integer, real or categorical, i.e., $\Theta_i \in \mathbb{R}$ or $\Theta_i \in \mathbb{Z}$ or $\Theta_i \in \{0, \ldots, c\}$, for some $c \in \mathbb{N}$, where c is the number of categories. A probabilistic model using the Gaussian Process finds the best configuration function f', which maps a configuration $\boldsymbol{\theta}$ to the estimated real-valued validation accuracy. This probabilistic model provides the properly balanced trade-off between exploitation and exploration of the search space [117]. At the end of the 1000 fixed trials the best configuration is obtained as $\theta^* = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} f'(\theta)$.

State-of-the-art AutoML Tools

In recent times, various AutoML systems or tools with complementary strengths have been proposed in the literature. A recent benchmark study [71] identifies AutoGluon [52, 53] as one of the leading AutoML systems, demonstrating superior performance across diverse datasets and tasks.

In contrast to traditional CASH-based approaches, introduced TabPFN, a general-purpose predictor trained across multiple datasets that can immediately deliver highly accurate predictions without extensive tuning [89, 90]. 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. Given their strong performance, AutoGluon and TabPFN are

selected to approximate the Bayes predictor for the proposed ILD approaches described in Section 3.2.2. The foundational concepts and strengths of these benchmark tools are outlined below.

AutoGluon In contrast to other AutoML systems that aim to select feature pre-processing algorithms and a learning algorithm and tune their respective hyperparameters, AutoGluon [52, 53] follows a different approach. More specifically, AutoGluon focuses on building a stacking ensemble, greedily adding more models as long as they are beneficial for the overall performance of the ensemble. To this end, AutoGluon searches over a wide range of different ML algorithms and pre-fitted models, including DL, gradient boosting, and linear models. Simultaneously, AutoGluon tunes hyperparameters for each model to optimize their performance. For performing HPO, different standard techniques can be used such as ASHA [116], Hyperband (HB) [117], Bayesian optimization (BO) [65], or Bayesian optimization and Hyperband (BOHB) [55]. A recent benchmark study found AutoGluon to perform superior to other AutoML systems in predictive performance and robustness. The only drawback noted by [71] is the prediction time, as it will grow with the number of models included in the returned stacking ensemble. Moreover, AutoGluon allows the customization of the set of considered ML algorithms and models, including gradient boosting machine (GBM) tree-based models and deep neural networks (NNs). For the experiments done for this thesis, the search space of AutoGluon is limited to the consistent classifiers [132, 18]. Table A.1 present the models and learning algorithms with corresponding hyperparameters, including their range values available to AutoGluon for the experiments in Chapters 4 and 5.

TabPFN TabPFN [89, 90] is an approach to AutoML that solves classification tasks in a transductive way. Instead of fitting a model specifically to a dataset and returning this model to the user, TabPFN is a transformer-based neural network architecture trained on the vast number of different probability distributions of tabular data. Instead of searching for a proper learning algorithm

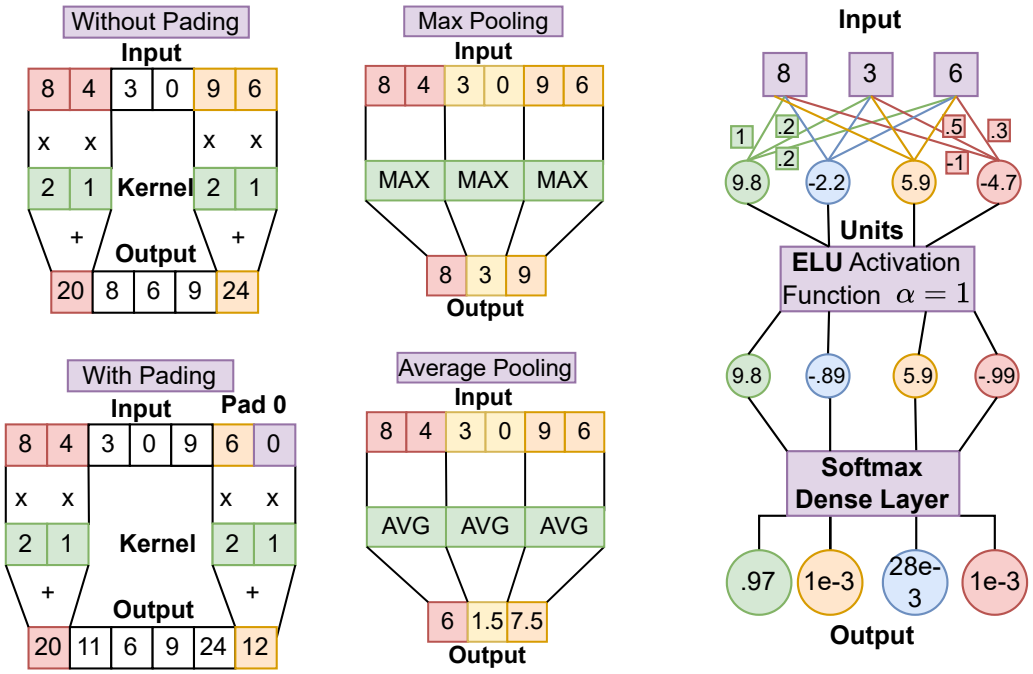
and its hyperparameter setting in the first place, TabPFN can be used to instantaneously make predictions as it is already fitted on prior data. Due to this, predictions can be obtained faster by orders of magnitude compared to classical AutoML estimation baseline tools. Moreover, the predictions are highly accurate and excel, particularly on small tabular datasets. However, TabPFN was only considered for datasets consisting of up to 1000 training data points, 100 numeric features, and 10 classes. Hence, in practical applications with more data points, TabPFN is currently not reliably applicable. Due to its highly accurate predictions, which appear to come with well-calibrated probability estimates for the classes, TabPFN is an interesting pick for the proposed ILD approaches to approximate the Bayes predictor. Since TabPFN is limited to handling datasets with up to 100 numeric features, the real-world IL-Datasets obtained from OpenSSL TLS server implementations contain more than 100 features. To address this, dimensionality reduction techniques, especially using RF and extra trees classifier (XT) (applied only when $d > 100$), are employed. Fine-tuning is conducted on TabPFN by adjusting parameters such as the number of reduced features, the reduction model, and the number of prior-fitted models through HPO.Table A.1 presents the specific parameter ranges used for the experiments detailed in Chapters 4 and 5.

Neural Networks

A Neural Network consists of a series of interconnected layers containing Neurons that connect an input layer that is activated according to observation with an output layer corresponding to the model's prediction for this observation. The structure of these interconnections and the method of layer operation can vary significantly and define the overall Neural Architecture. The MLPs is a very simple Neural Network, only employing fully connected, or "dense", layers. These were shown to perform SCA efficiently if the system applies no countermeasures but often fails for more challenging tasks [14, 120].

In recent work in the area of hardware side channels, the CNNs have proven to be very effective in learning a multi-class classification model and breaking a system via hardware side channels, even if such a system implements countermeasures [206]. CNNs have shown to be very robust towards the most common countermeasures, namely masking [120, 124] and desynchronization [23]. A CNN contains convolutional, pooling, and dense layers as shown in Figure 6.1. A CNN can be viewed as an MLP where only each neuron of the layer l is connected to a set of neurons of the layer $l-1$, therefore, can perform all the operations that can be performed by an MLP [206]. In addition to that, the CNN architecture imposes inductive biases that are useful for many critical applications and that the MLP networks would have to learn [206]. A recent study shows that, overall, CNNs are more efficient and better suited for performing SCA on hardware datasets than MLPs [25], which is why the focus is placed on different CNN architectures [79] for the experimental study in this thesis described in Chapter 6. The convolutional block consists of the convolutional layer and a pooling layer, and the dense block consists of the dense (fully connected layer). The batch normalization operation is typically applied after the convolutional and dense layers.Each layer has some trainable parameters $\hat{\boldsymbol{w}}$, which are used to get the final target function f (c.f. Section 2.3.3) and some hyperparameters. The hyperparameters are configuration variables of the layer external to the learning model (f) and hugely influence finding an optimal target function f. Next, the operations performed by these layers are briefly described.

Convolutional Operation This operation re-estimates the input value by taking a weighted average of the neighboring values as shown in Figure 2.4a. The weights are defined using a kernel of some size w_k (w_k for 1-D data or $w_k \times w_k$ for 2-D data), and these weights are learned using back propagation algorithm [206]. This kernel is shifted over the input data (1-D vector or 2-D maps) with a stride until the entire data is covered. The convolutional operation is performed for every shift and produces a weighted average value. Typically, this operation is applied multiple times using different kernels, and this number



(a) Convolutional Operation (b) Pooling Operations (c) Dense Layers

Figure 2.4.: Different operations of various CNN layers

is called the filter size f_i of the convolutional layer. If this operation is applied without padding, then the dimensionality of output decreases, which is called the valid padding operation. To preserve the dimension, the data is padded with 0, and this operation is called same padding [206].

The number of trainable parameters for convolutional layers is $[in \times f_i \times w_k \times out] + out$ for 1-D data and $[in \times f_i \times w_k \times w_k \times out] + out$ for 2-D data, where in denotes the number of inputs and out denotes number of outputs [169]. The two hyperparameters that need to be searched for an optimal CNN architecture are the kernel size and number of filters for each convolutional layer as listed in Table 6.1.

Pooling Operation This operation applies down-sampling on the input acquired from the previous layer and produces a condensed representation. This operation reduces the trainable parameters of the CNN and avoids over-

fitting [206]. The pooling operation of some size w_p (w_p for 1-D data or $w_p \times w_p$ for 2-D data) and stride is shifted across the input, reducing it by applying a max or average operation, as shown in Figure 2.4b. Similar to the convolutional operation, the pooling operation could also be applied with (preserves dimensionality) or without padding (dimensionality decreases), and the operations are called *same* or valid padding, respectively. This layer does not have any trainable parameters, and the hyperparameters that need to be searched for an optimal CNN architecture are the `pool-size` w_p, number of strides, and pooling operation type as listed in Table 6.1.

Dense Layers This layer consists of weights $\boldsymbol{W} \in \mathbb{R}^{d \times n_h}$ and biases $\boldsymbol{b} \in \mathbb{R}^{n_h}$, where d is the dimensionality of the input $\boldsymbol{x} \in \mathbb{R}^d$ and n_h is the number of hidden units of the layer [14]. The output of this layer is evaluated using the formulae $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ as shown in Figure 2.4c. Typically, an activation (e.g., `ReLU`, `Elu`) is also applied to each output element, and the weights and biases are learned using the back-propagation algorithm [14]. The last dense layer is applied using the softmax function (c.f. Section 2.3.3), which converts real-valued scores to *softmax-scores* as shown in Figure 2.4c. The number of trainable parameters for dense layers is the sum of n_h for each input plus the bias b: $n = (in \times n_h + n_h \times out) + (n_h + out)$ where in denotes the number of inputs and out denotes the number of outputs [169]. The hyperparameter that needs to be searched is the number of hidden units for each dense layer as listed in Table 6.1.

Batch Normalization Layer This layer was introduced to lower internal covariance shift in the neural network, thus making the convergence faster [95]. It was possible to use higher learning rates for the training process. This layer normalizes every data point $\boldsymbol{x}_i$ in a training batch by estimating the expected mean and the variance of the training batch. The number of trainable

parameters for batch normalization is $4 \times d$, where d is the dimensionality of the input. To perform the NAS, the operation is chosen to be applied to each convolutional and dense block, as depicted in Figure 6.1c.

2.3. Side-channel Attacks

This section provides a concise overview of SCAs, classifying them by the type of leaked information, collection methods, and proximity requirements for execution [211, 183]. It focuses on Bleichenbacher's SCA, which targets RSA-based encryption to extract plaintext, and template attacks that compromise secret keys in AES-encrypted systems, detailed in Section 2.3.2 and Section 2.3.3, respectively. As detecting ILs in these vulnerable systems is a central aim of this thesis, an understanding of these attack methods is essential for applying the ILD techniques presented in Chapter 3.

2.3.1. Taxonomy

A side channel is a mode through which unintended leakage of sensitive information occurs from a computing cryptographic device, including standard modes like network messages, CPU caches, power consumption, and expectation-maximization (EM) radiation [211]. A SCA uses the leaked information from one or multiple modes to reconstruct the original, sensitive data [211]. Such attacks have historical roots in espionage, such as NATO's TEMPEST and the Soviet PEMIN programs, which used EM emissions from cipher machines to decrypt diplomatic communications during the early Cold War [50].

Recent SCAs like GoFetch and SLUBStick have exposed significant vulnerabilities in modern systems. The GoFetch attack exploits data memory-dependent prefetchers (DMPs) in Apple CPUs to extract secret keys, emphasizing the need for more robust defenses against evolving SCAs, impacting cryptographic protocols like OpenSSL Diffie-Hellman, Rivest–Shamir–Adleman (RSA), posing

a significant threat to the macOS systems [29]. SLUBStick targets the Linux kernel leaking side-information in arbitrary memory, performing cross-cache SCAs, threatening cloud and container security environments [119]. According to Spreitzer et al. (2018) [183], SCAs are classified by method, mode or side channel used to gather the information, and attacker proximity, as shown in Figure 2.5.

Attack Methodology In **passive** SCAs, the attacker observes and collects information from side channels without interacting with or altering the device's operations and stealthily acquires sensitive information without changing the behavior of the device or introducing any faults into the system, making them hard to detect [211, 183]. **Active** SCAs involves inducing errors or faults via fault injection techniques like laser or clock glitching to reveal sensitive information from cryptographic devices [183]. This thesis focuses exclusively on noninvasive passive attacks, representing a significant threat in hardware and software environments.

Proximity of Attacker **Local** SCAs require direct physical access to the target device, such as measuring power consumption or EM emissions from smart cards [78, 177]. **Vicinity** SCAs exploit shared resources, like caches in multi-core processors, requiring proximity but not direct contact, timing-based attacks on shared caches [202] and analysis of WiFi and Bluetooth signals [208].

Remote SCAs can be executed over the network, exploiting processing time for the messages [130, 68] or cache behaviors without physical proximity [179]. For simplicity, the vicinity and remote attacks are merged under a single category, as both can be executed without physical access.

Side Channel Type **Physical (Hardware)** SCAs exploit the physical characteristics of devices, such as power consumption, electromagnetic emissions, or acoustic signals, by analyzing hardware-specific side channels like power

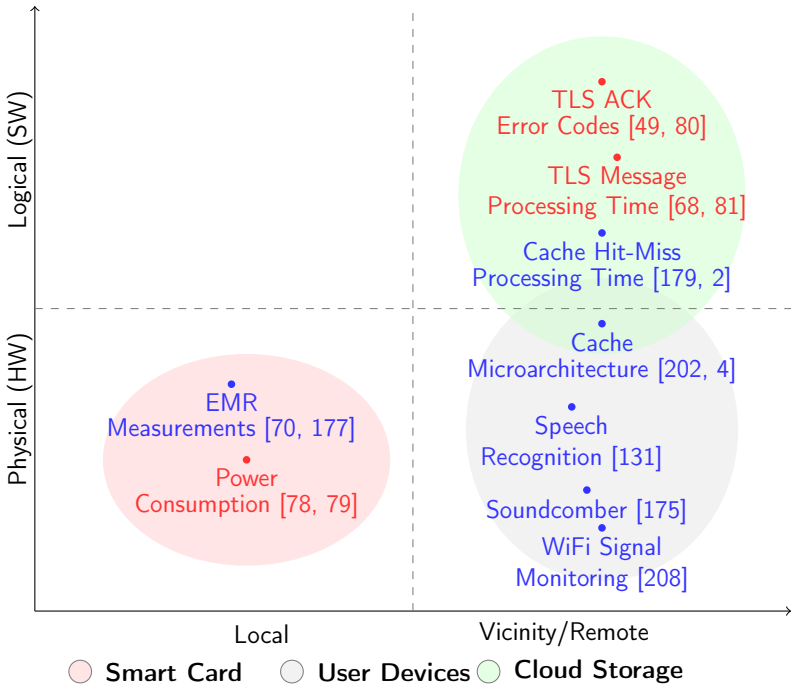


Figure 2.5.: Taxonomy of Side-channel Attacks [183]

and EM measurements to extract sensitive information from devices [78, 177, 175, 131]. **Logical (Software)** SCAs exploit the software layer by leveraging the logical execution behavior of software to carry out attacks [68, 49]. These attacks utilize processing time or error codes of the correctly and incorrectly padded PKCS#1 messages, which may leak through the system's cache or network traces, to extract sensitive information.

Relevant for Thesis This work addresses the detection of identity (ID) leakage in AES-encrypted systems, which are shown to be susceptible to **hardware** SCAs through **local** side channels such as power consumption or electromagnetic radiations (EMR) [79]. Additionally, ILs via the **remote** side channel of network traffic analysis are examined, where information is leaked through processing time, and alert responses (error codes) are Transport Layer Security (TLS)

handshake messages as shown in Figure 2.7 that distinguish between correct and incorrect plaintexts, rendering devices vulnerable to (**software**) Bleichenbacher SCAs [80, 81].

2.3.2. Bleichenbacher's Attack

Introduced by Bleichenbacher (1998) [20], Bleichenbacher's attack is a classic passive *remote* padding oracle SCA that exploits subtle server response differences, such as distinct TLS error messages or processing time variations, to recover sensitive information from RSA PKCS#1v1.5 encrypted ciphertexts. This attack involves sending carefully crafted ciphertexts repeatedly to a decryption oracle over the network, leveraging side information to distinguish valid from invalid padding string (PS) (e.g., PKCS#1v1.5). Ultimately, it allows the adversary to recover plaintexts or forge signatures, highlighting critical vulnerabilities in cryptographic implementations, making them a key consideration for designing robust IL detection and SCA defense strategies [20].

This section discusses the exploitation of padding errors in RSA-based encryption, as demonstrated by Bleichenbacher's attack, starting with an overview of the TLS handshake and attacker model, followed by the evolution of similar SCAs and Bleichenbacher's iterative algorithm for attacking TLS servers [49].

TLS Handshake and Authentication

TLS is the most widely used cryptographic protocol, securing web browsing, web applications, emails, VoIP calls, and virtual private network (VPN) traffic across most websites. Based on the Google Transparency Report, as of early 2023, 96 % of web pages visited by German Chrome users on Windows were loaded over HTTPS/TLS [76]. Known initially as Secure Sockets Layer (SSL), the protocol was renamed to TLS in 1999 [162], and the term TLS now encompasses all versions, including earlier SSL versions.

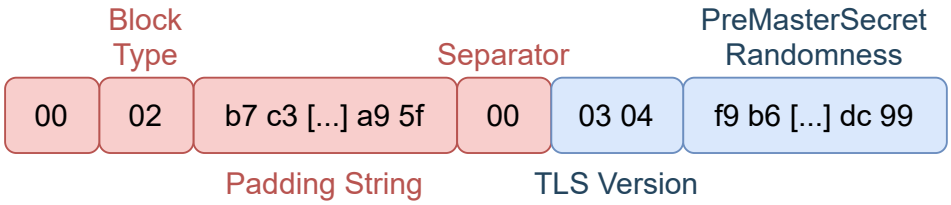


Figure 2.6.: Padded Pre-master secret (PMS) [163]

RSA-PKCS#1v1.5 Key Exchange All TLS cipher suites based on RSA key transport use the RSA-PKCS#1v1.5 encryption scheme for securely transmitting the pre-master secret (PMS) from the client to the server [163]. This scheme pads the PMS using a combination of constant and random bytes, structured as shown in Figure 2.6. The PMS is a random 48-byte string padded with: the leading `00 02`, a separator byte `00`, and the TLS version number `03 03` (indicating TLS 1.2). The remaining padding string is chosen randomly from all byte strings that exclude `00`, ensuring the padded string's total length matches the modulus size $\mathcal{N}$. The resulting padded string $\mathcal{P}$ is encrypted as $c = \mathcal{P}^e \bmod \mathcal{N}$.A ciphertext is **PKCS#1v1.5-conformant** if $\mathcal{P} = c^{1/e} \bmod \mathcal{N}$ follows the padding structure defined in Figure 2.6. Non-conformant ciphertexts are discussed in Section 5.1.

TLS defines a two-phase protocol: first, the handshake where the client and server authenticate each other and establish shared cryptographic keys, followed by the encrypted actual payload data transmission phase using the derived keys, as shown in Figure 2.7. RSA was the first key exchange algorithm used in TLS to secure shared secret transmission. In this scheme, the client generates a new PMS and encrypts it using the server's public RSA key. Once the server receives the encrypted PMS, it decrypts it using its private key, allowing both parties to derive the shared session keys for secure communication. Additionally, RSA is used for digital signatures in the server's certificate, enabling the client to verify the server's identity and guard against impersonation attacks.

The TLS 1.2 Handshake The TLS 1.2 handshake [166], illustrated in Figure 2.7, is the fundamental process for establishing a secure connection. During this handshake, the client and server agree on cryptographic algorithms, exchange secret keys, and the server authenticates its identity to the client. The sequence of handshake messages is as follows:

Step (1) `ClientHello` message The client initiates the connection by sending the `ClientHello` message, which contains a list of supported cryptographic algorithms (cipher suites).

Step (2) `ServerHello` message The server responds by selecting a cipher suite and sending the `ServerHello` message. It then proves its identity using a digital signature linked to a certificate from a trusted third party. The handshake flow ends with a `ServerHelloDone` message.

Step (3) ClientKeyExchange (CKE) The client proceeds with the key exchange using the agreed-upon algorithms. For RSA-based key transport, the client generates a new random key called the PMS, which is encrypted using the server's public RSA key, ensuring only the server can decrypt it. The client indicates the start of encrypted communication by sending a ChangeCipherSpec (CCS) message and a Finished (FIN) message containing a cryptographic checksum of the key exchange messages, computed using a key derived from the PMS.

Step (4) Server Verification and Finalization Upon receiving the encrypted PMS, the server decrypts it using its private key. It derives cryptographic keys to decrypt and verify the client's FIN message. The handshake concludes when the server sends a CCS and a final FIN message back to the client.

From this point onward, all communications between the client and server are secured using keys derived from the PMS and other publicly exchanged values (e.g., nonces in the `ClientHello` and `ServerHello` messages).

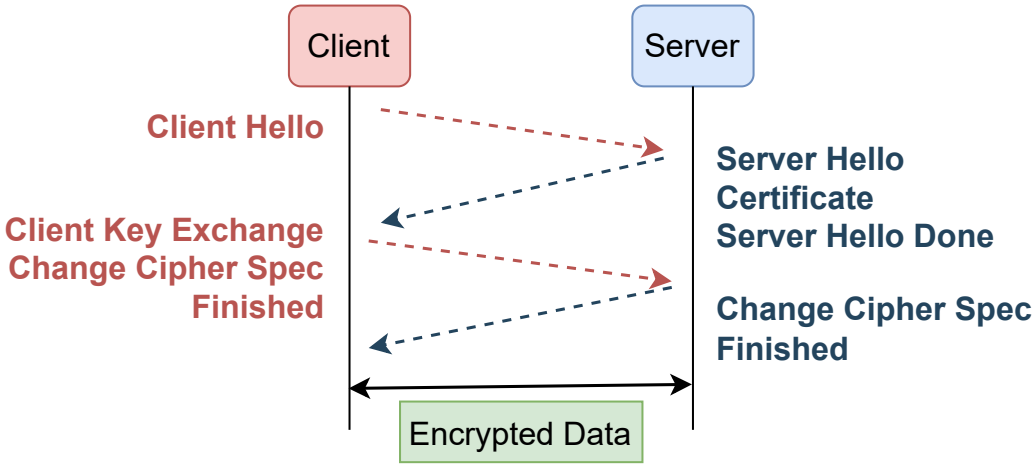


Figure 2.7.: TLS 1.2 Handshake example [166]

Evolution of Attacks

In Bleichenbacher's attack, the oracle takes an RSA-encrypted ciphertext as input. It reveals whether the decrypted plaintext is formatted according to the PKCS#1v1.5 padding standard, leaking *secret information* in the process. This requires a side channel within the protocol's implementation, such as a TLS server that returns distinct alert messages based on padding correctness. The attack leverages these discrepancies by sending manipulated ciphertexts in repeated TLS handshakes, observing the server's responses, and classifying them as "padding correct" or "padding incorrect". By accumulating these responses, the adversary can gradually reconstruct the original plaintext or generate valid RSA signatures with a few thousand queries. Understanding the separation between the *attack*, the *oracle*, the *side channel*, and the *implementation* is crucial, as eliminating the side channel in a specific implementation can mitigate the attack entirely.

Improvements to the oracle or attack techniques (e.g., the ROBOT enhancements [21]) can make previously hidden or un-exploitable side channels a new threat. Similarly, optimizations (such as those by Bardou et al. (2012) [6]) can turn impractical side channel vulnerabilities into feasible attacks. Numerous research papers have explored different methods for constructing and exploiting

this oracle in specific implementations or applications [20, 107, 96, 45, 209, 130, 5, 6, 58, 21, 102], revealing that such vulnerabilities frequently reappear in widely used software and commercial products. Consequently, Bleichenbacher-like attacks serve as a valuable reference for developing automated methods to detect SCAs on the protocol level, with TLS being a prime example [167].

Bleichenbacher's attack history Bleichenbacher's attack was first introduced by Bleichenbacher (1998) [20], targeting the RSA key exchange used in SSL version 3 [167]. Also known as the "million message attack" due to the high number of required TLS handshakes, it exposed a critical flaw in the PKCS#1v1.5 padding standard, which led to an update from the vulnerable padding in version 1.5 [163] to the optimal asymmetric encryption padding (OAEP) scheme in version 2 [164]. However, when TLS version 1.0 [162] was introduced as the successor to SSL version 3, it did not adopt the updated padding scheme. It continued to require PKCS#1v1.5 padding for RSA key exchange until TLS 1.2 inclusive, while TLS 1.3 removed the RSA key exchange with PKCS#1v1.5 [166]. The standard specified to address the issue by "The best way to avoid vulnerability to this attack is to treat incorrectly formatted messages in a manner *indistinguishable* from correctly formatted RSA blocks" [162, Section 7.4.7.1].

In 2003, Klíma et al. introduced a variant called the "bad version oracle", which exploited the rejection behavior of TLS implementations for invalid version numbers in RSA key exchange messages. By leveraging these known version bytes in the middle of the message instead of the initial bytes, they adapted Bleichenbacher's attack to use this new side channel [107]. This led to TLS version 1.1 [165, Section 7.4.7.1] and version 1.2 [166, Section 7.4.7.1] refining countermeasures to mask such side channels and standardizing uniform handling of invalid paddings. Specifically, TLS version 1.1 recommended suppressing TLS alerts for invalid version numbers, while TLS 1.2 introduced more comprehensive measures, standardizing the uniform handling of incorrect paddings to eliminate timing-based side channels. In 2018, new side channels were discovered when

Böck et al. [21] analyzed timeout behaviors in incomplete TLS handshakes, revealing vulnerabilities in major websites. Their analysis found that 27 of the 100 most visited websites (according to the Alexa ranking[1]) were susceptible to variations of Bleichenbacher's attack using these newly identified side channels. Although TLS version 1.3 [168] finally removed RSA key exchanges and the PKCS#1v1.5 padding, older versions of TLS remain in widespread use, leaving many servers exposed. According to the 2021 F5 Labs TLS Telemetry Report, 52 % web servers still allow the use of RSA key exchanges, even though the majority of servers prefer using elliptic curve key agreements like ECDHE, which aligns with Bleichenbacher's attack relevance [196]. This persistence in using outdated cryptographic practices has security implications, as highlighted by F5's ongoing tracking of top CVEs and DDoS trends, which show vulnerabilities in legacy protocols and their impact on attack strategies [83, 84].

Attacks based on Bleichenbacher's The original padding oracle attack described by Bleichenbacher (1998) [20] relied on distinguishable error messages in early TLS server implementations. Although later protocol versions introduced countermeasures, such as standardizing indistinguishable error responses, this did not eliminate all potential side channels. Over time, researchers have demonstrated that padding oracles can exploit various protocol behaviors and implementation-specific vulnerabilities. Klíma et al. (2003) [107] introduced the *bad version oracle*, leveraging timing variations in the TLS handshake to create a padding oracle.This marked a significant development in the evolution of Bleichenbacher's attack, as the vulnerability stemmed from subtle timing differences rather than easily distinguishable error messages. Jager et al. (2012) [96] demonstrated that XML Encryption, widely used in web services, could also be vulnerable to padding oracles. This attack exposed vulnerabilities at the application layer, highlighting that such attacks are not confined to the TLS protocol alone. In the hardware SCA realm, Bardou et al. (2012) [6] refined the original attack, drastically reducing the number of required oracle queries. They

[1] `https://www.alexa.com/topsites`

revealed padding oracles in several hardware tokens, including the Estonian ID card and RSA SecurID tokens, demonstrating that side channels can persist in specialized cryptographic hardware.

Degabriele et al. (2012) [45] extended these attacks to the Europay-Mastercard-Visa (EMV) payment protocol, demonstrating how padding oracles could compromise payment systems. Similarly, Meyer et al. (2014) [130] uncovered vulnerabilities in the Java Secure Socket Extension (JSSE) TLS implementation, identifying side channels that were detectable over a network. One of the most impactful advancements was the DROWN attack by Aviram et al. (2016) [5], which exploited a flaw in OpenSSL, allowing highly efficient Bleichenbacher-style attacks using outdated cryptography from SSLv2. This demonstrated that even deprecated protocols could be leveraged to compromise modern cryptographic systems. More recently, Felsch et al. (2018) [58] expanded the scope of these attacks by demonstrating padding oracles in the IPSec Internet Key Exchange (IKE) handshake, used by major vendors like Cisco and Huawei.

Zhang et al. (2014) [209] demonstrated that cache-based side channels in cloud environments could allow attackers to extract information across virtual machine boundaries, applying Bleichenbacher's attack to XML encryption on TLS servers in cloud hosting contexts. Jager et al. (2015) [97] showed that even protocols like TLS 1.3 and QUIC, which do not support vulnerable RSA key exchanges, could be impersonated using Bleichenbacher's attack on a different server supporting older versions like TLS 1.2. Xiao et al. (2017) [203] further expanded the attack by targeting a TLS library running in Intel's SGX enclave, using side channels to decrypt TLS handshakes executed by the library. In 2018, Böck et al. (2018) [21] reintroduced the threat of Bleichenbacher's oracle with the ROBOT attack, exposing new side channels through incomplete TLS handshakes. Ronen et al. (2019) [172] took advantage of microarchitectural side channels, particularly cache timing, to downgrade TLS handshakes and execute parallelized versions of Bleichenbacher's attack. Finally, Kelesidis (2021) [104] proposed optimizations that reduced the number of necessary oracle queries by 75 %, significantly improving the efficiency of the attack. These examples

highlight the ongoing relevance of Bleichenbacher's attack, emphasizing the need for constant surveillance and adaptation in cryptographic design to mitigate such SCAs.

Noisy side channels Noisy side channels introduce a significant challenge in SCAs by making extracting accurate information from the leaked data harder. Despite this, several techniques and attacks have been developed to effectively handle noisy observations, demonstrating that SCAs can succeed even in a noisy environment. For instance, Frittoli et al. (2020) [66] proposed strengthening sequential SCAs by using change detection tests to manage errors like false positives and false negatives. This method is effective in noisy environments as it identifies deviations in side channel leakage and corrects them, making attacks more resilient to inaccuracies. Similarly, Guo et al. (2020) [78] introduced Soft analytical side-channel attacks (SASCA), which applies coding theory to handle noisy leakage data, effectively exploiting noisy side channel information and making it robust across various attacks. Other notable works, such as Ronen et al. (2019) [172], addressed the challenge of false positives in noisy side channels by repeating oracle queries multiple times to enhance reliability, while Meyer et al. (2014) [130] examined the impact of false negatives on SCAs and showed how noise can affect the success of attacks. Moreover, some attacks have developed methods specifically to exploit noisy data. For example, Wu et al. (2022) [202] presented PREDATOR, a detection mechanism that remains effective in noisy environments by using precise event monitoring to detect microarchitectural manipulations.

Additionally, Ali and Khan (2022) [4] demonstrated that multi-core timing variations across shared hardware resources could be aggregated, making the attack more robust against noise than traditional single-channel attacks. Furthermore, the **Marvin SCA** by Kario (2024) [102] highlights how noise can be mitigated in timing-based SCAs. The attack uses paired difference tests and system optimizations, such as BIOS configuration, to maintain high CPU frequency and to detect even small timing differences in noisy environments. By refining

measurement techniques, the Marvin attack can bypass noisy observations and remain effective, underscoring how modern SCAs can adapt to and exploit noisy conditions. In summary, while noise can hinder SCAs, advanced techniques in error correction, statistical analysis, and system optimizations allow attackers to mitigate these challenges, making noisy side channels still vulnerable to exploitation and corresponding systems susceptible to SCAs.

Attack Methodology

This section details how Bleichenbacher's iterative algorithm leverages side-channel information to refine plaintext guesses until the complete message is recovered progressively.

Padding Oracle: Attacker Model Padding oracle attacks are a specific type of *remote* SCA that exploit the handling of padding within encryption schemes. During encryption, additional *padding* bytes are required to ensure that the resulting plaintext meets the block size requirements, particularly for block ciphers [106]. Adding randomized padding is also essential (but not necessarily sufficient) to be able to achieve IND-CPA secure encryption from deterministic public-key encryption schemes, like textbook RSA [106]. As such, PSs represents how the sender adds this data and how the receiver removes it. In these attacks, an oracle returns a binary response indicating whether the decrypted plaintext or the given ciphertext conforms to the padding standard. Although this may seem like a minor IL, attackers can repeatedly query the oracle with slight variations of the same ciphertext, systematically narrowing down the plaintext until only one possibility remains. This capability effectively allows attackers to decrypt a given ciphertext without access to the secret key or to forge digital signatures, as seen in Bleichenbacher's attack on PKCS#1v1.5 RSA padding [20], Vaudenay's attack on cipher block changing (CBC) padding [192], and Manger's attack on OAEP RSA padding [122]. This effectively allows the attacker to decrypt a given ciphertext without access to

the secret key or to forge a signature in the case of Bleichenbacher's attack on RSA. These attacks are only feasible when the ciphertexts are *malleable*, allowing predictable modifications to the plaintext. For example, RSA 's multiplicative homomorphism enables precise adjustments, while in the CBC mode, a modification of the initialization vector (IV) can predictably alter the first block.

Bleichenbacher's attacker model assumes an adversary with unlimited ability to interact with the implementation over the network, allowing them to send repeated messages indefinitely. This assumption is reasonable for many network protocols, where servers—such as web or mail servers—respond to requests persistently, making them vulnerable. Even traffic filtering mechanisms can be circumvented by attackers using distributed networks (e.g., botnets or VPNs) or sufficient patience. The side channel can include the primary protocol messages and metadata like Transmission Control Protocol (TCP) packets or timing information. In some cases, it is also assumed that the attacker has pre-existing access to certain ciphertexts, either by passively sniffing traffic on the network or through a man-in-the-middle (MitM) attack.

Attack Algorithm The Bleichenbacher's attack operates on the assumption that a *padding oracle* is available, which takes as input a ciphertext and reveals whether the contained plaintext conforms to the PKCS#1v1.5 PS for a given RSA public key $(\mathcal{N}, e)$. The attack uses this oracle only to decrypt RSA PKCS#1v1.5 ciphertexts but also compute valid RSA signatures with respect to the same RSA key contained in a server's certificate.

Bleichenbacher described a seminal algorithm that can use such an oracle to efficiently compute the RSA decryption function $ct \rightarrow ct^{1/e} \bmod N$ for any given ciphertext $ct \bmod N$. The oracle operates as follows:

$$\text{Oracle}(ct) = \begin{cases} 1 & \text{if } ct \text{ is PKCS\#1 conformant w.r.t.}(\mathcal{N}, e), \\ 0 & \text{otherwise.} \end{cases}$$

Such an oracle can be constructed through various side channels, such as analyzing error messages returned by a TLS server or detecting response timing differences.

The core idea of Bleichenbacher's algorithm is to leverage this padding oracle to decrypt RSA ciphertexts or compute valid RSA signatures. Essentially, the idea of Bleichenbacher's algorithm is as follows.

Suppose that $ct = \mathcal{P}^e \bmod \mathcal{N}$ is a PKCS#1-conformant ciphertext. This assumption holds without loss of generality, because if ct is not, the oracle can be used to randomize ct by computing $\hat{ct} = ct \cdot \rho^e = (\mathcal{P}\rho)^e \bmod \mathcal{N}$ for a random ρ, until $\hat{ct}$ is PKCS#1-conformant, and then continue with $\hat{ct}$.

The number or the PKCS#1-conformant message $ct\mathcal{P}^{1/e} \bmod \mathcal{N}$ lies in the interval $[2\mathcal{B}, 3\mathcal{B})$, where $\mathcal{B}$ denotes a number modulo $\mathcal{N}$ whose binary representation is:

$$\mathcal{B} = \texttt{00 01 ...00} = 2^{8(\ell-2)}$$

where ℓ is the byte-length of the modulus $\mathcal{N}$.

Step 1 Bleichenbacher's algorithm chooses a small integer s, computes:

$$\tilde{ct} = (ct \cdot s^e) \bmod \mathcal{N} = (\mathcal{P}s)^e \bmod \mathcal{N}$$

Step 2 If the padding oracle reveals that $\tilde{ct}$ has valid padding, then this implies:

$$2\mathcal{B} \leq \mathcal{P}s - r\mathcal{N} < 3\mathcal{B}\,,$$

for some integer r.

Step 3 This inequality is equivalent to the condition:

$$\frac{2\mathcal{B} + r\mathcal{N}}{s} \leq \mathcal{P} < \frac{3\mathcal{B} + r\mathcal{N}}{s}$$

Step 4 Thus, the plaintext $\mathcal{P}$ must lie within the interval:

$$\mathcal{P} \in \left[\lceil \frac{2\mathcal{B} + r\mathcal{N}}{s} \rceil, \lfloor \frac{3\mathcal{B} + r\mathcal{N}}{s} \rfloor \right).$$

Step 5 By repeatedly choosing new values of s, the algorithm constructs a set of intervals that progressively narrows down the possible values of $\mathcal{P}$ until only a single possibility remains, which must be the plaintext.

Bleichenbacher's algorithm's performance primarily depends on the precision of the oracle's response when checking the validity of the padding. Bardou et al. (2012) [6] described an improvement to Bleichenbacher's algorithm [6] and also analyzed its concrete efficiency for various types of oracles.

2.3.3. Template Attacks

Template attacks represent one of the most potent SCAs, where the adversary constructs detailed statistical models, called *templates*, based on the side channel of known cryptographic computations. In hardware security, template attacks exploit *local*, *physical (hardware)* side channels, such as power consumption or EMR, requiring proximity to the device, categorizing them as active local physical SCA, as shown in Figure 2.12. For example, Smart cards and trusted platform modules (TPMs), encrypted using the Advanced Encryption Standard (AES) described in Section 2.3.3 algorithm are particularly susceptible due to the feasibility of obtaining identical devices, making template attacks a key consideration for designing robust IL detection and SCA defense strategies.

AES Algorithm

The AES algorithm employs multiple rounds, where each round integrates the input with secret key material and thoroughly mixes the data, ensuring that dependencies on both input and key are propagated across all output bits [43].

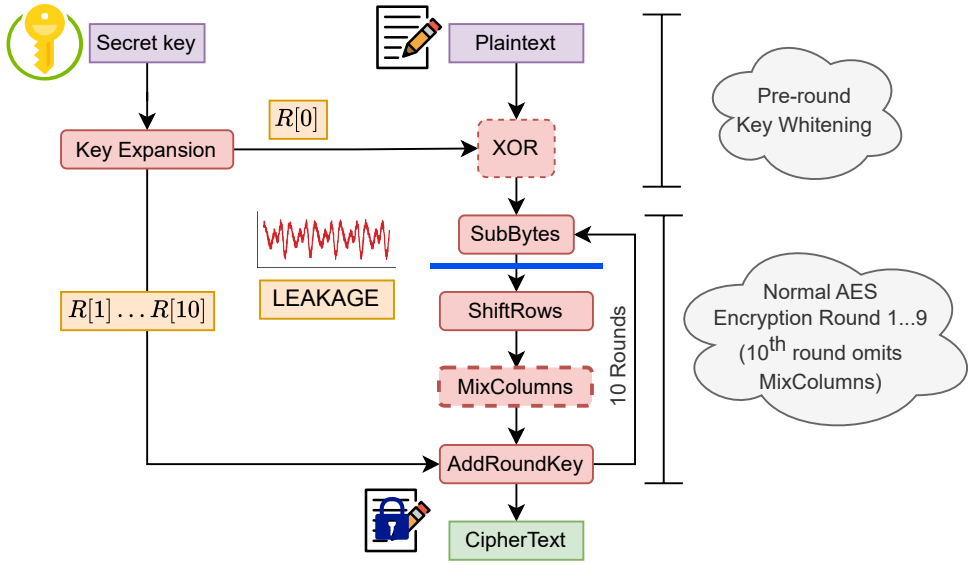


Figure 2.8.: Information leakage (IL) in AES Encryption Algorithm [43, 14]

AES operates on 256-bit cipher blocks arranged in a 4×4 byte matrix, in which plaintext, keys, and intermediate states are processed. It is constructed using a sequence of operations — `AddRoundKey`, `SubBytes`, `ShiftRows`, and `MixColumns`, combined with a critical schedule that generates a unique round key for each round, such that `MixColumns` step is omitted in the 10^{th} round, as illustrated in Figure 2.8. The decryption process in AES reverses the encryption steps, starting from the ciphertext and systematically applying inverse transformations to retrieve the original plaintext. This ensures that the transformations are undone exactly, mirroring the encryption sequence. Each round uses the inverse operations — `InvShiftRows`, `InvSubBytes`, `InvMixColumns`, and `AddRoundKey`, with the 10^{th} round omitting the `InvMixColumns` step, as shown in Figure 2.9.

Encryption The algorithm design ensures that it follows the fundamental properties of *confusion* and *diffusion* to obscure relationships between the plaintext, ciphertext, and key, as shown in Figure 2.8.

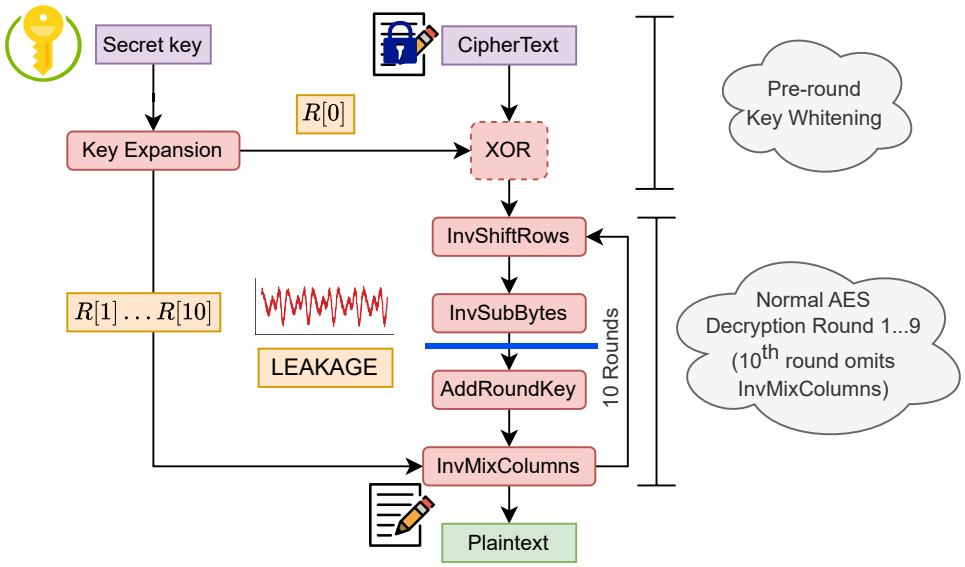


Figure 2.9.: Information leakage (IL) in AES Decryption Algorithm [43, 14]

Confusion maximizes the dependency of each ciphertext bit on multiple key bits, achieved using the `AddRoundKey` and `SubBytes` operations in AES. Diffusion ensures that changes in one input bit influence many output bits, making it difficult to isolate specific input changes Daemen and Rijmen (2002) [43]. In AES, `ShiftRows` and `MixColumns` are used to achieve this, adhering to the strict avalanche criterion, where a single bit change affects half of the ciphertext output [197]. The AES encryption process consists of several core operations applied in multiple rounds to maximize confusion and diffusion. *Key Expansion* generates a unique round key for each round from the initial cipher key using a key schedule function. The first `AddRoundKey` operation uses the cipher key directly, while subsequent `AddRoundKey` steps utilize derived round keys. The `AddRoundKey` combines the input state with the round key using the XOR operation, defined as $\texttt{AddRoundKey}(\boldsymbol{k}, \boldsymbol{tx}) = (tx_0 \oplus k_0, \dots, tx_{15} \oplus k_{15})$, where $\boldsymbol{k}$ is the round key and $\boldsymbol{tx}$ is the input state. The `SubBytes` operation substitutes each byte in the state matrix using a non-linear `S-Box` transformation, constructed from the multiplicative inverse in $GF(2^8)$ and an affine function, ensuring no values remain in their original position, an example of the `S-Box` transformation shown in Figure 2.10. The AES `S-Box` operation

SBox() itself is denotes as $\phi(t_i, k_i) = \text{SBox}(t_i \oplus k_i)$, where t_i is the i^{th} plaintext byte. `ShiftRows` cyclically shifts each row by varying offsets to propagate

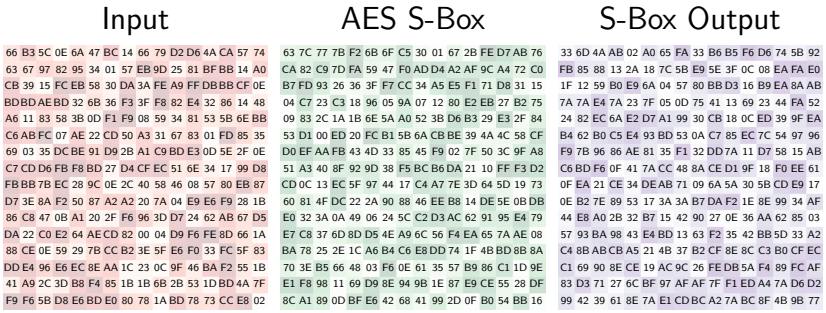


Figure 2.10.: `S-Box` for AES Encryption Algorithm

byte dependencies across columns, and `MixColumns` replaces each byte in a column with a linear combination of all column values, further spreading the dependencies across the matrix. When applied sequentially across multiple rounds, these operations destroy statistical patterns, ensuring that even small changes in the key or plaintext result in significant differences in the ciphertext output.

Decryption The decryption process in AES reverses the transformations performed during encryption by systematically applying the inverse operations — `InvSubBytes`, `InvShiftRows`, `InvMixColumns`, and `AddRoundKey`, as shown in Figure 2.9. Each round works backward, starting from the ciphertext and reintroducing the original plaintext bit-by-bit by reversing the non-linear substitutions and the mixing operations Daemen and Rijmen (2002) [43]. Like encryption, Key Expansion generates a unique round key for each step using the same key schedule function. In contrast to the encryption procedure, the decryption process first applies the `AddRoundKey` operation using the round key, proceeding with `InvSubBytes`, which performs a byte-wise substitution using the `Inverse S-Box`, constructed as the inverse of the standard `S-Box` in $GF(2^8)$. As shown in Figure 2.10, input is passed through the `S-Box` to acquire the encrypted output, which, when passed through the `Inverse S-Box`, recovers the input back, as shown in Figure 2.11. The AES `S-Box` operation SBox()

itself is denotes as $\phi^{-1}(c_i, k_i) = \text{SBox}^{-1}(c_i \oplus k_i)$, where c_i is the i^{th} ciphertext byte. Each inverse operation is designed to undo the changes made during encryption, ensuring that dependencies between plaintext and ciphertext bits are exactly reversed. Next, `InvShiftRows` shifts each row in the matrix to its

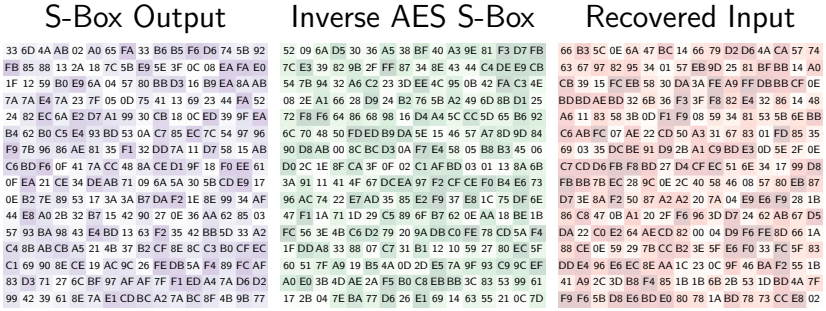


Figure 2.11.: `Inverse S-Box` for AES Decryption Algorithm

original position, undoing the byte transpositions caused by the `ShiftRows` step. Finally, `InvMixColumns` operates on each column by applying the inverse of the MixColumns matrix multiplication, reversing the linear combination of bytes. The 10^{th} round omits the `InvMixColumns` step, similar to the encryption, to maintain the final round's structure and ensure the decryption's integrity. This omission ensures that the byte arrangement and value propagation from the encryption's last round are exactly restored, maintaining a consistent plaintext output. The overall process ensures that all confusion and diffusion introduced during encryption are precisely reversed, recovering the original plaintext from the ciphertext.

Identity Leakage

In AES-based systems, secret information can be revealed via power traces measured during the `SubBytes` operation. This results in the leakage of the complete output (e.g., ID) of the intermediate `S-Box` transformation rather than just partial information (e.g., hamming weight (HW)). The ID leakage model assumes that each unique intermediate value within the algorithm corresponds to a distinct and identifiable leakage pattern. For AES, this is most evident

at non-linear transformations like the `S-Box`, where the intermediate state $v = \phi(t_i, k_i) = \text{SBox}(t_i \oplus k_i)$ is directly linked to the power consumption or EMR leakage of the device. Each of the 256 possible `S-Box` outputs produces a unique leakage pattern, enhancing the signal-to-noise ratio (SNR) and making correct key guesses easily distinguishable, which makes the ID leakage model highly suitable for profiling SCAs [157, 198].

Leakage via S-Box Operation In AES encryption, leakage is observed after the `S-Box` operation, represented as $v = \phi(t_i, k_i) = \text{SBox}(t_i \oplus k_i)$, where t_i is the plaintext byte and k_i is the corresponding key byte. Instead of correlating the key directly with the measured leakage, the ID model targets the intermediate value (e.g., i^{th} plaintext byte $\phi(t_i, k_i)$), making it unnecessary to know the `S-Box` lookup table explicitly. Here, side channel traces form distinct patterns, allowing attackers to match leakage with specific S-box outputs. This step, shown in Figure 2.8, is particularly vulnerable due to the non-linear nature of the `S-Box`, making it a high-risk point for SCAs.

Leakage via Inverse S-Box Operation During AES decryption, leakage occurs at the inverse `S-Box` operation, modeled for each byte as $\text{SBox}^{-1}(c_i \oplus \text{RoundKey}_{10,i})$, $\forall i \in [15]_0$, where c_i is the i^{th} byte of the ciphertext and $\text{RoundKey}_{10,i}$ is the corresponding byte from the round 10 key. Unlike encryption, this scenario uses a known-ciphertext attack, where the attacker profiles the leakage at the `Inverse S-Box` step in the initial round, making it an equally attractive target for SCAs, as shown in Figure 2.9. Observing the variations in the power consumption or EMR emissions during the inverse transformations makes it possible to distinguish between various key byte candidates, as the correct key guess will produce distinct leakage patterns.

Generalization The `S-Box` transformation is applied independently to each byte of the AES state, allowing attacks to be partitioned into 16 smaller, byte-wise attacks: $\text{SBox}(\boldsymbol{t} \oplus \boldsymbol{k}) = (\text{SBox}(t_0 \oplus k_0), \ldots, \text{SBox}(t_{15} \oplus k_{15}))$ [74]. This approach significantly reduces complexity by focusing on one byte at a time, making the attack more efficient and more accessible to implement [74].

Countermeasures

Countermeasures for SCAs can be broadly categorized as either **hiding** or **masking** [27]. These techniques aim to decouple the measurable leakage from the secret by reducing the observable impact of data on the side channel or by randomizing the intermediate values to obscure the secret information.

Hiding and Masking **Hiding** works by reducing how much a value or computation influences measurable factors such as power consumption or EMR. This is typically achieved by altering the device's construction or randomizing the execution order of operations. The objective is to make the measurable leakage independent of the processed data. However, even with precise design, some residual leakage is often unavoidable [26]. While hiding can minimize data-related leakage, it does not eliminate the risk of SCAs, as sophisticated techniques could still exploit the remaining leakage.

Masking introduces an additional random hidden state into the computation, which is later removed to reveal the correct output. Unlike hiding, masking does not reduce the influence of intermediate values on the side channel; instead, it randomizes those values, revealing less about the secret to potential attackers. A well-known example of this approach is **Boolean masking**, where a random value (or mask) is XORed with the input and then removed later during the computation [109]. Masking makes the intermediate data independent of the secret, effectively obscuring the sensitive information. However, the success of masking depends heavily on generating sufficient entropy in the random values used as masks.

Weaknesses and Vulnerabilities Despite the effectiveness of masking as a countermeasure, it has weaknesses. Masking assumes that attackers cannot recover the random mask values. Obtaining sufficient entropy for masking can be challenging, especially in hardware implementations where dedicated hardware entropy generators significantly increase system complexity and power consumption. Moreover, the masking process may become the target of SCAs. For example, attackers can attempt to recover the mask value and use this knowledge to attack the actual secret [150]. Thus, even though masking offers significant protection, it is not foolproof.

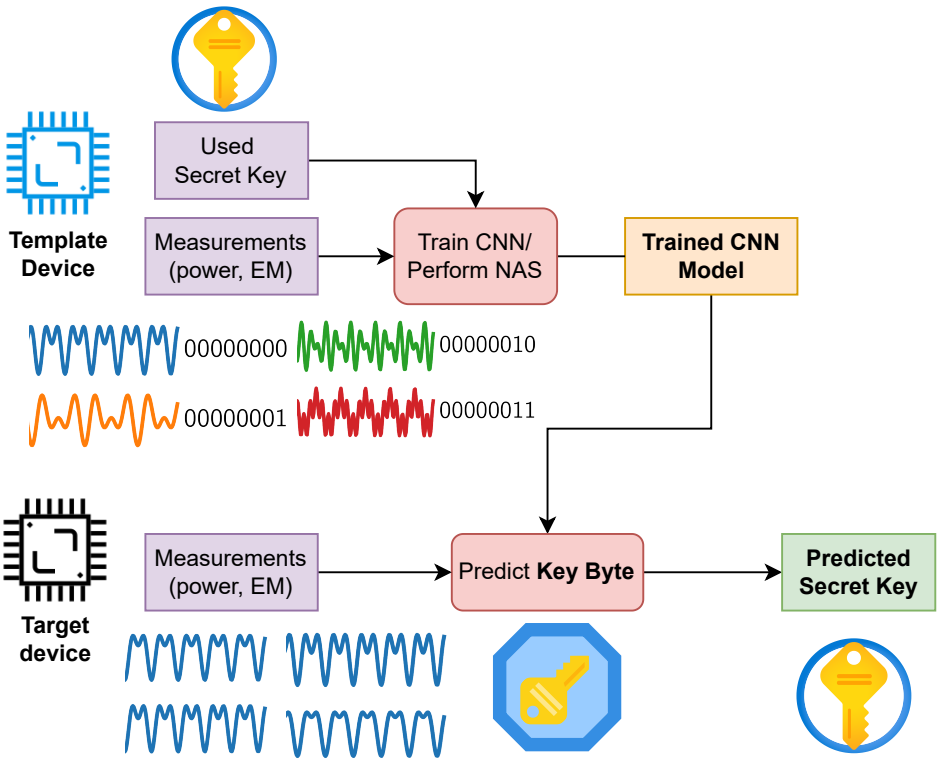


Figure 2.12.: DL-based Template SCA

DL Based SCA In systems with **first-order leaks**, where the secret directly influences the measured side channel, statistical countermeasures like masking are generally effective [109]. However, introducing uncorrelated hidden states,

such as in Boolean masking, can lead to **higher-order leaks**, which are more challenging to exploit and require more advanced techniques and larger data sets [26, 129]. DL has become a powerful method for detecting side channel leakage, particularly in the case of **higher-order leaks**, where traditional ML models struggle [120, 14, 92]. Its ability to handle complex masked systems has made DL the preferred approach for countermeasure-resistant SCAs. This thesis focuses on DL-based SCAs targeting AES, though the techniques are also applicable to other cryptographic systems.DL is especially effective at identifying vulnerabilities in masked systems, offering more robust analysis than traditional methods.

Recent Countermeasures In response to these advanced attack strategies, Chari et al. (1999) [26] developed a theoretical model to quantify how much masking is required to protect a system from adversaries. Their concept of **higher-order masking** defends against attackers capable of observing multiple data paths simultaneously.Despite this, systems using **Boolean masking** have been successfully attacked, prompting researchers to explore more robust methods, such as **affine masking** [199, 67]. Affine masking provides more robust protection by adjusting intermediate values to minimize leakage. This method's effectiveness is demonstrated in the **ASCADv2 dataset** [14], which combines affine and Boolean masking, and to date, no successful attacks have been reported against it [150].

Attack Methodology

In a SCA, an attacker aims to determine the secret key used in a cryptographic operation, e.g., an encryption process running on a target device. The attack methodology of exploiting the communication channel capacity to reveal the secret key of the system is illustrated in Figure 2.12. For non-profiled attacks, the attacker observes the device without access to the private key, relying on measurements such as EMRs or power consumption [150].

Often, it is assumed that the attacker has access to a second, identical device, termed a *template device*, to execute the attack via *profiling* mechanism [150]. This approach, known as profiled SCA, involves the attacker building a behavioral profile by running cryptographic operations with known secret keys on the *template device*, as shown Figure 2.12. The attacker collects N observation traces, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, during the profiling phase, each corresponding to different secret key bytes used in the AES algorithm. These traces are time-series measurements of power consumption or EMRs, represented as d-dimensional real-valued vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ for $i \in [N]$. In the attack phase, this profile is employed to recover the secret key from the observed behavior of the target device, as depicted in Figure 2.12.

Communication Channel Capacity The SCA exploits the capacity of the communication channel to extract information about the secret key K from observed leakage traces.In this setup, the secret key $K = (k_0, \ldots, k_{15}), \forall k_j \in \mathcal{K}$, and the plaintext $T = (t_0, \ldots, t_{15}), \forall t_j \in \mathcal{T}$, are processed by the device to produce intermediate values $y_i \in \mathcal{Y}$, such as the output of an S-box in AES, represented by Y. The information about Y is leaked through side channels like power consumption or EMRs, via N traces, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, represented by the random variable X, which contains crucial information about K, as shown in Figure 2.13.

The **encoder** represents the process by which the device combines the secret key K with the plaintext $\boldsymbol{t}_i = (t_0, \ldots, t_{15}), \forall t_j \in \mathcal{T}$, represented by the random variable T, to produce intermediate values $y_i \in \mathcal{Y}$, represented by Y, and how this information leaks through power or EMRs. The **channel** introduces noise from the remaining parts of the device or imperfections in the measurement setup. In contrast, the **decoder** is the attacker's distinguishing rule, which uses the N measured traces $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and the known plaintext $\boldsymbol{t}_i = (t_0, \ldots, t_{15}), \forall t_j \in \mathcal{T}$ to infer the secret key $\hat{K}$.

This process follows a Markov chain $(K,T) \rightarrow (Y,T) \rightarrow (X,T) \rightarrow (\hat{K})$, where the intermediate values $y_i \in \mathcal{Y}$ are produced during encryption, such that $I((K,T);(X,T)) \leq I((Y,T);(X,T))$ The conditional MI $I(X;Y|T)$ quantifies the information leaked about the secret key K through the traces $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, represented by the random variable X, given the known plaintext T. By exploiting this leakage, the SCA recovers the secret key $\hat{K}$ by utilizing the communication channel's capacity to reduce uncertainty about K [31].

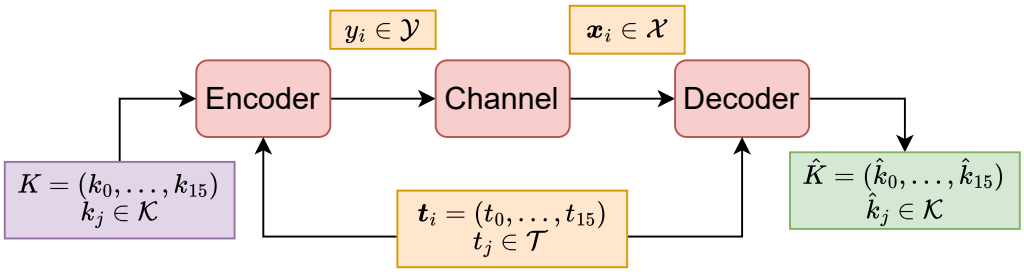


Figure 2.13.: Information Flow: $(K,T) \rightarrow (Y,T) \rightarrow (X,T) \rightarrow (\hat{K})$ [31]

Profiling The attacker records traces $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from the *template device*, such that each of these N traces corresponds to a known secret key byte $k_i \in \mathcal{K}$, with $\mathcal{K} = \{0, \ldots, 255\}$, and a known plaintext byte t_i. If different keys are used, the key bytes $k_1, \ldots, k_N$ are different in each trace, while $k_1 = k_2 = \ldots = k_N = k$ if the same key is used. Each profiling trace is labeled with $y_i = \phi(t_i, k_i)$ using a function ϕ, which maps the plaintext t_i and key k_i to a value that is assumed to relate to the deterministic part of the measured leakage $\boldsymbol{x}_i$ [148]. This mapping depends on the assumed ID leakage model and is typically defined using the AES S-box operation: $y_i = \phi(t_i, k_i) = \text{SBox}(t_i \oplus k_i)$.This labeling results in the profiling dataset $\mathcal{D}^{\text{Prof}} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, which is used by a supervised learning algorithm to build a profiling model.

Supervised Learning Template SCA The goal is to predict the secret key byte k_i used in the cryptographic operation from the attack traces $\boldsymbol{x}_i$, for which the key is unknown. The profiling dataset $\mathcal{D}^{\text{Prof}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$ is used, with $\mathcal{X} = \mathbb{R}^d$ as the input space (measured traces) and $\mathcal{Y} = \{0, \ldots, M-1\}$ as the output space, representing the 256 possible labels produced by $\phi(t_i, k_i)$.Here, $M = 256$ corresponds to the possible key byte values.

Then, the probabilistic function g_p is learned by minimizing the empirical risk. This function allows the aggregation of probabilities over multiple traces. Typically, g_p is learned by estimating a scoring function $h_{s|\boldsymbol{\theta}}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^M$, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$ [19, chap. 4]. The MLP outputs un-normalized scores $h_{s|\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{s} = (s_0, \ldots, s_{M-1})$, where $s_m = h_{s|\boldsymbol{\theta}}(\boldsymbol{x})[m]$ is the score for class m.

The *softmax* function converts these scores into probabilities $\hat{\boldsymbol{p}} = (\hat{p}_0, \ldots, \hat{p}_{M-1})$:

$$S_{\boldsymbol{\theta}}(\boldsymbol{x})[y] = \frac{\exp(s[y])}{\sum_{m=0}^{M-1} \exp(s[m])} = \frac{\exp\big(h_{s|\boldsymbol{\theta}}(\boldsymbol{x})[y]\big)}{\sum_{m=0}^{M-1} \exp\big(h_{s|\boldsymbol{\theta}}(\boldsymbol{x})[m]\big)} \,.$$

The empirical risk minimizer g_p is obtained by minimizing the CCE loss:

$$R_{\text{emp}|p}(h_{s|\boldsymbol{\theta}}) = \frac{1}{N} \sum_{i=1}^{N} -\ln(S_{\boldsymbol{\theta}}(\boldsymbol{x}_i)[y_i]) \,.$$

Attack Dataset and Key Recovery In the attack phase, the attacker records N_a attack traces, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_a}$, by sending plaintexts (or ciphertexts) $t_1, \ldots, t_{N_a}$ to the target device. Each trace corresponds to an unknown key byte $k^* \in \mathcal{K}$ and a known plaintext t_i. The attacker considers every possible key byte candidate $k \in \mathcal{K}$. For each instance $(\boldsymbol{x}_i, t_i)$, labels are generated for each $k \in \mathcal{K}$ using the same function $\phi(t_i, k)$ as during profiling. The resulting labels are denoted $\boldsymbol{y}_i = (y_{i,0}, \ldots, y_{i,M-1})$, such that $y_{i,k} = \phi(t_i, k)$.

The attack dataset is $\mathcal{D}_a^{\text{Att}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), \ldots, (\boldsymbol{x}_{N_a}, \boldsymbol{y}_{N_a})\}$, which the learned profiling model uses to recover the secret key byte k^*. The scoring function $S_{\hat{\boldsymbol{\theta}}}$ predicts scores for every key byte candidate $k \in \mathcal{K}$. For each attack instance

$(\boldsymbol{x}_i, \boldsymbol{y}_i)$, the scores of every key byte candidate are denoted as

$$\hat{\boldsymbol{s}}_i = (S_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i)[y_{i,0}], \dots S_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i)[y_{i,M-1}]) = (\hat{s}_{i,0}, \dots, \hat{s}_{i,M-1}) \,,$$

where $\hat{s}_{i,k} = S_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}_i)[y_{i,k}]$ represents the score of key byte candidate k [14].

The cumulative score function for each key byte k is calculated using the maximum log-likelihood as

$$\boldsymbol{d}^{N_a}[k] = \sum_{i=1}^{N_a} \log(\hat{s}_{i,k}) = \sum_{i=1}^{N_a} \log(\hat{\boldsymbol{s}}_i[y_{i,k}]) \,. \tag{2.11}$$

Using the likelihood to acquire the cumulative scores is an outlier-sensitive operation, as a single low score value can completely disqualify the true key [118]. To enhance the robustness and minimize sensitivity to low scores, the attack is executed R_a times on shuffled traces from the attack dataset N_a, resulting in the corresponding cumulative scores $\boldsymbol{d}^{N_a}[k]$. Using this, the ranking vector denoted by $(\boldsymbol{r}^{N_a} = (r_0, \dots r_{255}))$ containing the rank of each candidate key byte $\forall \hat{k} \in \mathcal{K}$ is defined. In this context, the rank $(r_{\hat{k}} = \boldsymbol{r}^{N_a}[\hat{k}])$ of the candidate secret key byte $\hat{k} \in \mathcal{K}$ is defined as

$$\boldsymbol{r}^{N_a}[\hat{k}] = 1 + \left(\sum_{k \in \mathcal{K} \setminus \hat{k}} [\![\boldsymbol{d}^{N_a}[k] > \boldsymbol{d}^{N_a}[\hat{k}]]\!] \right) \,. \tag{2.12}$$

The vector is indexed with superscript N_a, to represent the result from performing the a^{th} attack out of R_a repeated attacks with attack dataset $\mathcal{D}_a^{\text{Att}}$, each containing N_a traces, as $\boldsymbol{r}^{N_a}$.

Evaluation Metrics

In the context of SCAs, evaluating the effectiveness of an attack relies on specific metrics that quantify the ability of the attacker to recover the secret key. Three widely used metrics are guessing entropy (GE), trace sufficiency threshold

(TST), and success rate (SR), which are commonly applied in profiling and non-profiling attacks [123, 184, 169]. These metrics help assess how quickly and efficiently a SCA can be performed to acquire the secret key of the system.

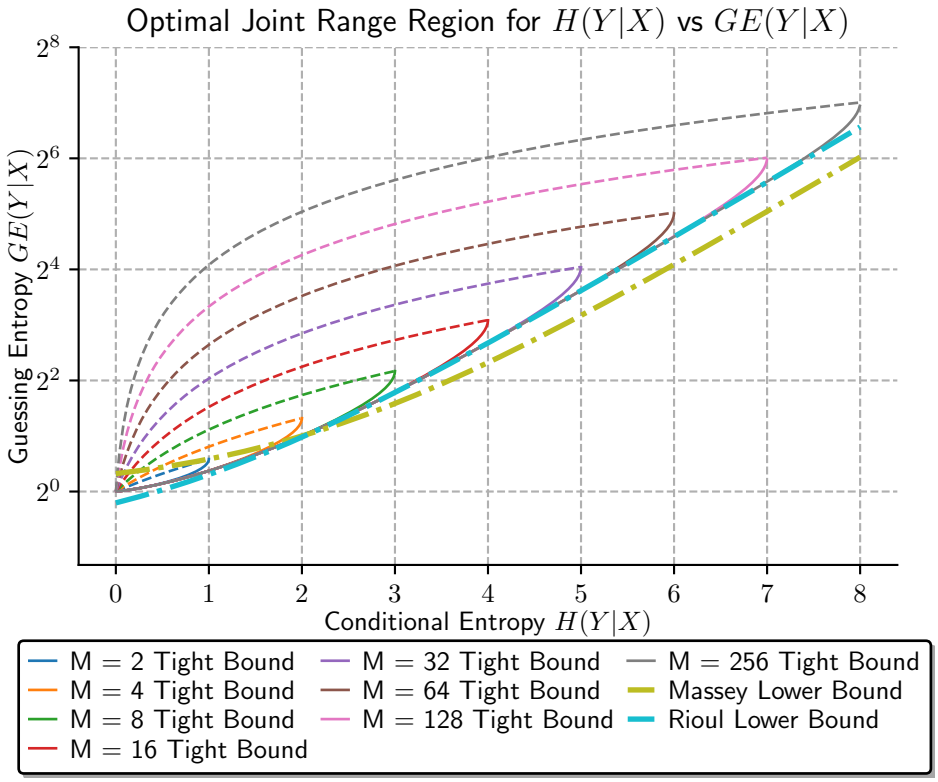


Figure 2.14.: Optimal joint range between $H(Y|X)$ and GE $G(Y|X)$ for classes M in range $[2, 256]$, with upper bound derived by McEliece and Yu (1995) [125] (dashed) and lower bound by Béguinot and Rioul (2024) [11] (solid), with the sub-optimal lower bound by Massey (1994) [123] and Rioul (2022) [171] is M-independent

Guessing Entropy The GE is a measure of the expected number of guesses required for an attacker to recover the correct key byte k^* [123]. The GE for the correct key byte k^* using the ranking vector $\boldsymbol{r}^{N_a}[\hat{k}]$ in eq. (2.12) is estimated as

$$\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*) = \boldsymbol{r}^{N_a}[k^*] \approxeq \frac{1}{R_a} \sum_{a=1}^{R_a} \boldsymbol{r}^{N_a}[k^*]\,, \tag{2.13}$$

By averaging GE values across repeated R_a experiments, the overall security in terms of the actual GE, denoted by $G(Y|X)$, can be estimated accurately.

Relation to Conditional Entropy The relationship between GE and conditional entropy $H(Y|X)$ is critical for understanding how much information is revealed by the side channel, for which several bounds help to formalize this relation, as illustrated in Figure 2.14. The sub-optimal lower bound independent of M is derived by Massey (1994) [123] and Chérisey et al. (2019) [31] as

Massey (1994) [123] Lower Bound

$$2^{H(Y|X)-2} + 1 \leq G(Y|X)$$

Chérisey et al. (2019) [31] Improved Lower Bound

$$\frac{2^{H(Y|X)}}{e} + \frac{1}{2} \leq G(Y|X)$$

Then McEliece and Yu (1995) [125] derived a bound on $G(Y|X)$ in terms of $H(Y|X)$ as

$$2^{H(Y|X)-1} + 1 \leq G(Y|X) \leq \frac{H((Y|X))(M-1)}{2\lg(M)} + 1 \tag{2.14}$$

However, the lower bound is only applicable when $G(Y|X) \geq 2$. Additionally, Béguinot and Rioul (2024) [11] derived a gamma-based parametric **lower bound** for different M values as

$$H(Y|X) = \log_2\left(\gamma \cdot \frac{1-\gamma^M}{1-\gamma}\right) - \log_2(\gamma) \cdot G(Y|X) \tag{2.15}$$

$$G(Y|X) = \frac{\log_2\left(\gamma \cdot \frac{1-\gamma^M}{1-\gamma}\right) - H(Y|X)}{\log_2(\gamma)},$$

where γ is a parameter used to express the functional relation between $H(Y|X)$ and $G(Y|X)$ in a parametric form, ranging in $\gamma \in (0, 1)$. The parameter γ induces the truncated geometric distribution, which optimally captures the trade-off between $H(Y|X)$ and $G(Y|X)$. As $\gamma \to 0^+$, the conditional entropy $H(Y|X)$ approaches zero, and the guessing entropy $G(Y|X)$ reaches its minimum value of 1, indicating complete IL. Conversely, for $\gamma = 1$, the uniform distribution is achieved, where $H(Y|X) = \log(M)$ and $G(Y|X) = \frac{M+1}{2}$, reflecting no IL. The parametric bound, therefore, allows for capturing different levels of IL by varying γ within the range $(0, 1)$, making it particularly effective for analyzing small ILs in SCAs [11].

Trace Sufficiency Threshold The TST indicates the minimum number of attack traces required for the model to guess the correct system secret key byte k^*, which is determined based on the ranking vector as $Q_{t_{GE}} = \arg\min_{n_a \in [N_a]} \boldsymbol{r}^{n_a}[k^*] = 1$ [169]. When the available N_a attack traces are insufficient, the value is set to N_a for aggregation in experiments. By averaging these values across repeated R_a experiments, the overall security efficiency in terms of the TST can be estimated accurately as

$$\mathrm{m}_{\mathrm{TST}}(\boldsymbol{r}^{N_a}, k^*) = \begin{cases} Q_{t_{GE}} & \text{if } \boldsymbol{r}^{N_a}[k^*] = 1, , \\ N_a & \text{if } \boldsymbol{r}^{N_a}[k^*] > 1. \end{cases} \approxeq \frac{1}{R_a} \sum_{a=1}^{R_a} \mathrm{m}_{\mathrm{TST}}(\boldsymbol{r}^{N_a}, k^*) \tag{2.16}$$

Success Rate The SR is the probability that the correct key byte k^* is ranked first, given a fixed number N_a of attack traces [184], which is formally defined as

$$\mathrm{m}_{\mathrm{SR}}(\boldsymbol{r}^{N_a}, k^*) = p[\boldsymbol{r}^{N_a}[k^*] = 1] \approxeq \frac{1}{R_a} \sum_{a=1}^{R_a} [\![\boldsymbol{r}^{N_a} = 1]\!] \tag{2.17}$$

The SR is an accuracy metric that evaluates how frequently the correct key can be recovered given N_a traces, providing insights into the SCAs's performance [188, 118]. The actual SR, denoted as $P_{\mathrm{SR}}(K|Y)$, is estimated by

calculating the percentage of instances where the rank of the actual key byte k^* is successfully determined across repeated R_a attacks, thereby quantifying the extent of IL.

Relation to Guessing Entropy Béguinot et al. (2022) [10] derived the bound on GE with respect to the SR as

$$(1+\lfloor\frac{1}{P_{\text{SR}}(K|Y)}\rfloor)(1-\lfloor\frac{1}{P_{\text{SR}}(K|Y)}\rfloor\frac{P_{\text{SR}}(K|Y)}{2}) \leq G(Y|X) \leq 1+\frac{M}{2}(1-P_{\text{SR}}(K|Y))$$

The GE is approximately inversely proportional to the SR, i.e., $P_{\text{SR}}(K|Y) \propto \frac{1}{G(Y|X)}$ particularly for small values of SRs, $P_{\text{SR}}(K|Y) = 1$ and large number of classes $M \geq 2^8$, implying that a higher SR corresponds to lower GE, reflecting an easier guessing process [11].

Relation to Mutual Information The upper bound on MI in terms of SR, showing that greater side channel information increases the likelihood of exactly guessing the key by an adversary, is derived by Chérisey et al. (2019) [31] as

$$H(K) - H_2(P_{\text{SR}}(K|Y)) - (1 - P_{\text{SR}}(K|Y)) \log_2(2^n - 1) \leq I(X;Y|T)\,,$$

where $H_2(a) = -a\lg(a) - (1-a)\lg(1-a)$ is the binary cross-entropy function of the SR, and when $a = P_{\text{SR}}(K|Y) = 1$, the MI $I(X;Y|T)$ reaches its maximum to $\lg(M)$, corresponding to complete leakage of the key.

2.4. Statistical Tests

This section explains the statistical tests used in the proposed ILD approaches described in Section 3.2.2. These include two types of Student's t-tests, namely the one-sample t-test (OTT) and paired t-test (PTT), and the Fisher's exact test (FET). To enhance **robustness and reliability** in ILD, the Holm-Bonferroni

correction is applied to account for multiple statistical tests conducted on various top-performing models or pipelines. This approach mitigates overfitting and reduces noisy estimates from relying on a single model [91].

2.4.1. Student's t-tests

Student's t-test is a statistical method for assessing whether the means of one or more groups differ significantly from a known value or each other. It includes the one-sample t-test (OTT), which compares the sample mean to a known population mean, and the paired t-test (PTT), which compares the means of paired or related samples to evaluate significant differences between them [46].

One-sample t-test

The OTT is used to determine if the mean of a sample differs significantly from a known expected or hypothesized population mean [46]. This test evaluates if sample values significantly deviate from an expected mean, like zero. The vector $\boldsymbol{a} = (a_1, \ldots, a_K)$, represents the collected $K = 10$ sample data points or observations. Next the sample mean ($\mu_{\boldsymbol{a}} = {}^1\!/\!_K \sum_{k=1}^{K} a_k$) and the standard deviation ($\sigma_{\boldsymbol{a}} = \sqrt{{}^1\!/\!_{(K-1)} \sum_{k=1}^{K} (a_k - \mu_{\boldsymbol{a}})^2}$) computed using the vector $\boldsymbol{a}$. Using the evaluated sample mean and standard deviation, the t-value representing the standard error from the expected mean under the null hypothesis is calculated as

$$t = \frac{(\mu_{\boldsymbol{a}} - \mu_0)}{{}^{\sigma_{\boldsymbol{a}}}\!/\!_{\sqrt{K}}},$$

where μ_0 under $H_0(\cdot)$ and $K = 10$ is the sample size. Then, the t-distribution with $K - 1$ degrees of freedom is analyzed to calculate the p-value, representing the probability of observing the sample mean if the null hypothesis is true. The null hypothesis $H_0(\mu_{\boldsymbol{a}} \sim \mu_0)$ implies no significant difference from the population mean (0 in case of IL detection), and the alternative hypothesis $H_1(\mu_{\boldsymbol{a}} \gg \mu_0 || \mu_{\boldsymbol{a}} \ll \mu_0)$ indicates a significant deviation from the mean. In the

case of quantifying and detecting IL, only the case ($\mu_{\boldsymbol{a}} \gg 0$ is considered since the IL in a system is always greater than 0 (IL$\in \mathbb{R}^+$). If the p-value is below a predefined significance level (e.g., $\alpha = 0.01$), the $H_0(\cdot)$ is rejected, indicating a significant deviation from the population mean μ_0; otherwise, the test fails to reject $H_0(\cdot)$ and accept the $H_1(\cdot)$. The one-sample t-test (OTT) assumes a normal population distribution, independent observations, and a sufficient sample size for the t-value to approximate the t-distribution.

Paired t-test

PTT is used to compare two paired samples (generated from the same underlying population) where each observation in one sample is matched with an observation in the other sample [46]. Consider two K-sized vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, representing the paired observations.

The null hypothesis is $H_0(\boldsymbol{a} = \boldsymbol{b})$, which assumes no significant difference between the paired samples, while the alternate hypothesis $H_1(\boldsymbol{a} \neq \boldsymbol{b})$ implies that there is a significant difference between the two samples. The p-value indicates the probability of obtaining the observed mean difference between the two samples, assuming that the null hypothesis $H_0(\cdot)$ holds, meaning the observations are drawn from the same distribution or have nearly zero average difference [46].

The t-value is computed as

$$t = \frac{\mu}{\sigma/\sqrt{K}},$$

where $\mu = \frac{1}{K}\sum_{k=1}^{K} d_k$ represents the mean difference between the paired values $d_k = a_k - b_k$, and $\sigma^2 = \sum_{k=1}^{K} \frac{(\mu - d_k)^2}{K-1}$ is the sample variance.

Nadeau (2003) [138] proposed a correction to account for dependency in estimates due to K-fold cross-validation (KFCV), adjusting the variance as:

$$\sigma_{Cor}^2 = \sigma^2 \left(\frac{1}{K} + \frac{1}{K-1} \right).$$

This corrected variance is used to recalculate the t-value as $t = \frac{\mu}{\sigma_{Cor}}$. The p-value, derived from the area under the Student's t-distribution curve, represents the probability of accepting H_0.

If the p-value is below a predefined significance level (e.g., $\alpha = 0.01$), the null hypothesis H_0 is rejected, indicating a significant difference between $\boldsymbol{a}$ and $\boldsymbol{b}$; otherwise, the test fails to reject the H_0. However, PTT relies on the assumption of normal distribution and asymptotic behavior, which can sometimes result in overly optimistic p-values. Additionally, accuracy measures can be misleading for imbalanced datasets [154, 148]. Moreover, the PTT requires a sufficiently large K for precise results, and a small test set can affect the correction term $\frac{1}{K-1}$, leading to less reliable estimates.

2.4.2. Fisher's Exact Test

Fisher's exact test (FET) is a non-parametric test that evaluates the independence between two classification methods by analyzing the contingency table [64]. For example, in a sample population classified by gender and cricket preference, if men predominantly like cricket and women do not, FET would yield a low p-value, indicating a correlation between the two classification methods. In the context of binary classification, FET uses a 2×2 contingency table, i.e., the CM) to assess the independence of ground truth labels $\boldsymbol{y}$ and classifier predictions $\hat{\boldsymbol{y}}$. The CM comprising of true positives (m_{TP}), true negatives (m_{TN}), false positives (m_{FP}), and false negatives (m_{FN}) in eq. (2.10), which, is used to compute the exact p-value via the hypergeometric distribution defined as

$$p(\boldsymbol{y}, \hat{\boldsymbol{y}} | \boldsymbol{M}) = \frac{C^{(R)}_{(\mathrm{m_{TN}})} \times C^{(N-R)}_{(r-\mathrm{m_{TN}})}}{C^{(N)}_{(r)}} = \frac{C^{(\mathrm{m_{FN}}+\mathrm{m_{TN}})}_{(\mathrm{m_{TN}})} \times C^{(\mathrm{m_{FP}}+\mathrm{m_{TP}})}_{(\mathrm{m_{FP}})}}{C^{(\mathrm{m_{TP}}+\mathrm{m_{TN}}+\mathrm{m_{FP}}+\mathrm{m_{FN}})}_{(\mathrm{m_{TN}}+\mathrm{m_{FP}})}}$$

where $r = \mathrm{m}_{\mathrm{TN}} + \mathrm{m}_{\mathrm{FP}}$, $N = \mathrm{m}_{\mathrm{FP}} + \mathrm{m}_{\mathrm{TP}} + \mathrm{m}_{\mathrm{FN}} + \mathrm{m}_{\mathrm{TN}}$, and $R = \mathrm{m}_{\mathrm{TN}} + \mathrm{m}_{\mathrm{FN}}$ and the binomial coefficients $C^{(n)}_{(r)}$ represent the number of ways to choose r items from n items.

The p-value is computed by summing the probabilities $p(\boldsymbol{y}, \hat{\boldsymbol{y}}|\boldsymbol{M})$ for all contingency tables with the same marginal counts as the observed and having a probability less than or equal to that of the observed table ($\boldsymbol{M}$). This approach evaluates the likelihood of obtaining a table at least as "extreme" under the null hypothesis of independence. 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 the two classification methods (e.g., classifier predictions and ground truth labels) are independent, implying no significant relationship. In contrast, 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, which could indicate a relationship between the two classifications. FET is especially suited for small sample sizes or imbalanced datasets, where approximation-based tests like the χ^2 test might fail to provide accurate results. It is commonly used for general classification tasks, particularly when comparing the performance of two classifiers based on their confusion matrices.

Advantages in Detecting Leakages The mathews correlation coefficient (MCC) is another evaluation measure related to FET, often preferred over accuracy due to its ability to handle imbalanced datasets [32]. MCC ($\mathrm{m}_{\mathrm{MCC}}$) is a balanced accuracy evaluation measure that penalizes m_{FP} and m_{FN} equally and accounts for imbalance in the dataset [32]. Camilli (1995) [24] demonstrated that MCC is directly proportional to the square root of the χ^2 statistic, i.e., $|\mathrm{m}_{\mathrm{MCC}}| = \sqrt{\chi^2/N}$, which is asymptotically equivalent to FET. This makes FET a more reliable method for detecting ILs when dealing with imbalanced datasets, compared to paired tests like PTTs, which may lead to overly optimistic results due to their reliance on normality assumptions. In summary, FET provides a rigorous and exact method for assessing the dependence between two classification methods, making it well-suited for both general classification tasks and specific ILD evaluations.

2.4.3. Holm-Bonferroni Correction

The Holm–Bonferroni correction controls the family-wise error rate, minimizing false positives (type-1 errors) by adjusting the rejection criteria α for each hypothesis within the family of null hypotheses, $\mathcal{F} = \{H_1, \ldots, H_J\}$, ensuring that the significance level of $\mathcal{F}$ not exceeding predefined threshold of $\alpha = 0.01$ [91]. Our process begins by independently testing each of the J models or pipelines, resulting in J p-values $(p_1, \ldots, p_J)$, sorted in ascending order, and making an aggregated decision. For each hypothesis $H_j \in \mathcal{F}$, if its associated p-value p_j is less than $\alpha/J+1-j$, the null hypothesis is rejected. This continues until $p_{\tau+1} > \alpha/J-\tau$, with rejected hypotheses represented by $\mathcal{F}_r = \{H_1, \ldots, H_\tau\}$, and the remaining hypothesis (non-rejected) by $\mathcal{F}_a = \{H_{\tau+1}, \ldots, H_J\}$.

The number of rejected hypotheses, denoted as $\tau = |\mathcal{F}_r|$, serves as the cut-off parameter and reflects the confidence in IL detection decisions. A higher value of τ indicates greater confidence in the detection outcome. To detect IL, a *rejection threshold* is set on the cut-off parameter τ quantifying IL detection confidence, corresponding to the number of rejected hypotheses, $|\mathcal{F}_r|$. 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. This systematic and rigorous approach empowers us to detect and characterize IL with a high degree of confidence, ensuring the robustness of the ILD framework.

3. Information Leakage Detection

This chapter introduces the framework for the information leakage detection (ILD) problem, utilizing the concepts of Bayes predictor and mutual information (MI) detailed in Section 2.1 and Section 2.2, respectively, with notations from Section 1.3. A generalized leakage assessment score (LAS) measure is presented to quantify information leakage (IL) with its variants applied to detect IL in a system as described in Section 3.2.1. Section 3.3 introduces Mid-Point, Log-Loss, and state-of-the-art MI estimation methods, employed to detect ILs in OpenSSL TLS servers using the one-sample t-test (OTT) as detailed in Chapter 5. To quantify ILs in AES-encrypted systems with secret key bytes, a specialized GE-based LAS measure, vulnerability score (VS), directly correlated to MI, is proposed Section 3.2.2. Along with trace sufficiency threshold (TST), it evaluates the susceptibility of AES-encrypted systems to template side-channel attacks (SCAs), detailed in Chapter 6. This chapter builds on work published in Gupta et al. (2024) [81].

3.1. Problem Formulation

ILD algorithm aims to identify unintended disclosure of *secret information* through *observable information* of the system, by analyzing the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}, N \in \mathbb{N}$ generated by the system, where $\mathcal{X} = \mathbb{R}^d$ is the observable information and $\mathcal{Y} = [M]$ secret information as categorical classes.

The goal is to label $\mathcal{D}$ with 1 indicating IL and 0 its absence, represented by the 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. This 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 standard binary classification metrics ($\mathrm{m}_{(\cdot)}(\boldsymbol{z}, \hat{\boldsymbol{z}})$), described in Section 2.2.2.

Evaluation Metrics

The performance of an ILD approach inducing the $\hat{L}$ function is assessed using Accuracy, FPR and FNR with respect to ground-truth vector $\boldsymbol{z}$ and predicted IL decisions $\hat{\boldsymbol{z}}$, which are defined in Section 2.2.2.

Accuracy It is the proportion of correct predictions and is defined as

$$\mathrm{m}_{\mathrm{ACC}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) := \frac{1}{N} \sum_{i=1}^{N} [\![z_i = \hat{z}_i]\!] .$$

False Positive Rate It is the ratio of *false positives* to the number of negative instances and defined as

$$\mathrm{m}_{\mathrm{FPR}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) = 1 - \mathrm{m}_{\mathrm{TNR}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) = \frac{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{z}, \hat{\boldsymbol{z}})}{\mathrm{m}_{\mathrm{FP}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) + \mathrm{m}_{\mathrm{TN}}(\boldsymbol{z}, \hat{\boldsymbol{z}})} .$$

False Negative Rate It is the ratio of *false negatives* to the number of positive instances and defined as

$$\mathrm{m}_{\mathrm{FNR}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) = 1 - \mathrm{m}_{\mathrm{TPR}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) = \frac{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{z}, \hat{\boldsymbol{z}})}{\mathrm{m}_{\mathrm{FN}}(\boldsymbol{z}, \hat{\boldsymbol{z}}) + \mathrm{m}_{\mathrm{TP}}(\boldsymbol{z}, \hat{\boldsymbol{z}})} \,.$$

3.2. Methodology

The complete methodology for performing ILD is described by first presenting the generalized measure, referred to as LAS, for quantifying IL in a system. Subsequently, the different methods used to detect IL in the system are introduced, which estimate various variants of LAS, namely MI, Bayes error rate ($\mathrm{m}_{\mathrm{ER}}\,(g^{bc})$), and guessing entropy (GE), as detailed in Section 3.2.1.

3.2.1. Leakage Assessment Score

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 $\boldsymbol{x}$ [105, 28]. To quantify the IL, the LAS is evaluated by comparing the (approximate) performance of the Bayes predictor and marginal Bayes predictor on the dataset generated by the system. When using log-loss, it reduces to MI commonly used for quantifying the IL in the given system.

The LAS $\delta(\cdot)$ evaluating the difference in average penalties of marginal Bayes predictor and Bayes predictor using loss functions ($\ell_{(\cdot)}$) or metrics ($\mathrm{m}_{(\cdot)}$), is defined as

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

where $|\cdot|$ is used to avoid negative values for accuracy measures. Note, that for log-loss loss function, LAS is equal to **MI**, i.e., $\delta(\ell_{ll}) = I(X;Y) = \mathbb{E}[\ell_{ll}(g^{mc})] - \mathbb{E}[\ell_{ll}(g^{bc})]$.

In practice, since $p_{(X,Y)}(.)$ is seldom observed, the LAS is approximated using empirical risk minimizers (g_p in eq. (2.8) or g in eq. (2.5)) as proxies for Bayes predictor and g_p^{mc} in eq. (3.3) for marginal Bayes predictor as

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

For losses evaluated using (conditional) class probabilities, g_p minimizing eq. (2.8) is used as a proxy for the Bayes predictor. However, g minimizing eq. (2.5) or the decision rule of g_p is used for classification decision-based losses, with g being a better proxy for error rate or accuracy measure.

Detection IL is considered to occur if LAS is significantly greater than 0, i.e., $\delta(\mathrm{m}_{(\cdot)}) \gg 0$ or $\delta(\ell_{(\cdot)}) \gg 0$. Thus, ILD is conducted by analyzing the learnability of empirical risk minimizers (g_p or g) on the system dataset $\mathcal{D}$. Statistical tests on multiple estimates of LAS are employed to detect IL in a system. ILD approaches are introduced using an average error rate (0-1 loss) to quantify IL by evaluating $\delta(\mathrm{m}_{\mathrm{ER}})$ (or $\delta(\mathrm{m}_{\mathrm{ACC}})$) and $\delta(\mathrm{m}_{\mathrm{MCC}})$ (via confusion matrix (CM)) and detecting IL using paired t-test (PTT) and Fisher's exact test (FET), respectively, as detailed in Section 3.2.2. MI-based ILD approaches are proposed, utilizing the OTT on multiple MI estimates, as described in Section 3.2.2, with Log-Loss effectively detecting IL in OpenSSL TLS servers, discussed in Section 5.3. For ILs in AES-encrypted systems, where the secret information is a secret key byte, a specialized GE-based LAS measure is proposed. This measure is directly correlated to MI, as shown in Section 3.2.2, and is used to derive a VS for the secret key byte of the AES-encrypted system, detailed in Chapter 6.

3.2.2. Approaches

The ILD approaches are categorized based on the use of MI and the classification performance of the Bayes predictor to detect IL in a system, referred to as **MI-based** and **classification-based** ILD approaches, respectively. The auto-

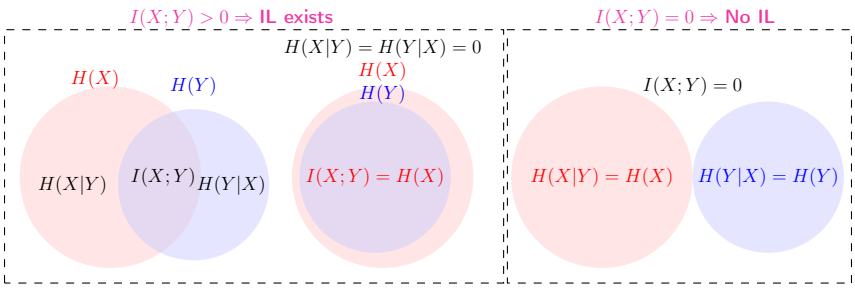


Figure 3.1.: IL-Quantification using MI

mated machine learning (AutoML) tools, AutoGluon and TabPFN, described in Section 2.2.4, are employed to accurately estimate empirical risk minimizers (g or g_p) as proxies for the Bayes predictor. Statistical tests are applied to the LAS evaluated using Log-Loss, accuracy, and CM of the top-performing AutoML models or pipelines (g or g_p) to obtain a p-value with a predefined significance level of $\alpha = 0.01$. If the p-value is below α, H_0 is rejected, indicating the presence of IL; otherwise, H_0 is not rejected. For detecting ILs in AES-encrypted systems, where the sensitive information comprises secret key bytes, a specialized GE-based LAS measure is employed, which is directly correlated to MI, as detailed in Section 3.2.2.

Mutual Information: Quantification and Detection

According to information theory, IL occurs in a system when the MI between system inputs and outputs is significantly greater than 0. This condition forms the basis for detecting IL in the given system. Figure 3.1 illustrates IL quantification using MI, a non-negative and symmetric measure that aligns well with the nature of IL in cryptographic systems. This makes it a robust metric for quantifying IL, as discussed in Section 2.1.2. The leftmost diagram in Figure 3.1 represents $I(X;Y) > 0$, indicating the presence of IL in the system. The middle diagram shows the maximum possible IL, with the MI value reaching $\min(\{H(X), H(Y)\}) = H(Y)$, signifying full dependence. Finally, the rightmost diagram depicts $I(X;Y) = 0$, indicating no IL in the system.

The LAS is equivalent to **MI** when using the log-loss loss function to evaluate the performance of Bayes predictor and marginal Bayes predictor. The Bayes predictor and marginal Bayes predictor is estimated using empirical risk minimizers $(g_p,\ g)$ and g_p^{mc}, respectively, i.e., $\delta(\ell_{ll}) = I(X;Y) = \mathbb{E}[\ell_{ll}(g^{mc})] - \mathbb{E}[\ell_{ll}(g^{bc})]$, marking as a foundation for the Log-Loss MI estimation approach described in Section 3.3.2. Additionally, the Bayes error rate $(\mathrm{m}_{\mathrm{ER}}\,(g^{bc}))$ bounds MI, forming the basis of the Mid-Point MI estimation approach in Section 3.3.1. To ensure accurate MI estimation, the use of AutoML tools such as AutoGluon and TabPFN, as discussed in Section 2.2.4, is recommended. This section details the application of OTT to the MI estimates obtained through Log-Loss and Mid-Point techniques for performing ILD in a cryptographic system.

One-sample t-test Based Approach As described in Section 2.4.1, the one-sample t-test (OTT) is used to determine if the mean of a sample differs significantly from a known population mean [46]. The MI-based ILD approach is founded 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 3.2.1. In the context of detecting IL in the system, the OTT is applied to evaluate whether the MI estimates obtained using K-fold cross-validation (KFCV) deviate significantly from zero. To deduce the presence of IL in a system, using the OTT, first consider the null hypothesis $(H_0(\hat{\boldsymbol{I}}_j \sim 0))$ and the alternative hypothesis $(H_1(\hat{\boldsymbol{I}}_j \gg 0))$. The null hypothesis (H_0) posits that the MI values are approximately zero, indicating no IL. In contrast, the alternative hypothesis (H_1) suggests a significant deviation from zero, implying the presence of IL. The $K = 10$ MI estimates are computed through KFCV of the top-j^{th} performing pipeline acquired using the Log-Loss and Mid-Point techniques and the baseline methods, represented by the vector $\hat{\boldsymbol{I}}_j = (\hat{I}_1, \ldots, \hat{I}_{10})$. Using them, the sample mean $(\mu_{\hat{\boldsymbol{I}}_j})$ and standard deviation $(\sigma_{\hat{\boldsymbol{I}}_j})$ is calculated. The t-statistic is computed as $t = \frac{\mu_{\hat{\boldsymbol{I}}_j} - \mu_0}{\sigma_{\hat{\boldsymbol{I}}_j}/\sqrt{K}}$, where $\mu_0 = 0.0$ is the expected population mean

under the null hypothesis and K is the number of MI estimates in the sample (in this case, 10). The p-value is derived from the t-distribution with $K-1$ degrees of freedom to evaluate the significance of the calculated t-statistic.

If the null hypothesis is true, the p-value represents the probability of observing a sample mean as extreme as the 0 obtained. If the p-value is below a predefined threshold (e.g., $\alpha = 0.01$), H_0 is rejected, indicating the presence of IL. Otherwise, H_0 is not rejected, and the alternate hypothesis H_1 is accepted, suggesting no significant IL. In summary, the OTT is applied to the MI estimates to assess whether they deviate significantly from the expected population mean of 0 to detect potential IL in the system. Subsequently, the corresponding ILD approaches using the MI estimation methods detailed in Section 3.3 combined with OTT to detect ILs in systems are referred as **MI-based** approaches in Chapter 5, Appendix A.2 and A.3.

Classification Performance: Quantification and Detection

This section outlines two ILD methodologies introduced in Gupta et al. (2022) [80], which are based on the concept from statistical learning theory stating that IL occurs in a system if the accuracy of Bayes predictor significantly surpasses that of a marginal Bayes predictor, as explained in Section 2.2.1. These methodologies quantify IL by analyzing the LAS using average error rate (measured by 0-1loss), i.e., $\delta(\mathrm{m}_{\mathrm{ER}})$, to quantify the performance difference between Bayes predictor and marginal Bayes predictor, as discussed in Section 3.2.1.

An accuracy-based measure, such as 0-1loss or average error rate, can be misleading for imbalanced datasets. To address this, the $\delta(\mathrm{m}_{\mathrm{MCC}})$, measuring the performance difference between the Bayes predictor and marginal Bayes predictor using mathews correlation coefficient (MCC), is analyzed through CM. FET is applied to the CM to assess the statistical dependence between ground truth and predicted labels, effectively incorporating $\delta(\mathrm{m}_{\mathrm{MCC}})$ analysis for detecting IL, especially in imbalanced datasets.

Gupta et al. (2024) [81] proposes using advanced AutoML tools like AutoGluon and TabPFN (c.f. Section 2.2.4) to evaluate the empirical risk minimizer g in eq. (2.5), improving upon Gupta et al. (2022) [80], which relied on a predefined set of binary classifiers to induce Bayes predictor. Leveraging AutoML ensures better adaptability across datasets and more reliable IL detection due to its consistent learners like multi-layer perceptron (MLP) and ensemble methods (random forest classifier (RF), gradient boosting machine (GBM)). These approaches are referred to as **Classification-based** in Chapter 5, Appendix A.2 and A.3.

Paired t-test Based Approach Our first approach detects IL in a system by using PTT to compare the accuracy estimates of marginal Bayes predictor using g_p^{mc} in eq. (3.3) and AutoML pipelines, which approximate the performance of Bayes predictor in eq. (2.6) using empirical risk minimizer g minimizing eq. (2.5). In this approach, the PTT is applied between the $K = 10$ accuracy estimates obtained through KFCV for the j^{th} best-performing AutoML pipeline estimating g (empirical risk minimizer) and g_p^{mc}, denoted by $\boldsymbol{a}_j$ and $\boldsymbol{a}_{mc}$, respectively. This approach, which evaluates whether the performance of marginal Bayes predictor significantly differs from Bayes predictor, is denoted as **PTT-Majority** [80]. A baseline approach proposed in Drees et al. (2021) [49] is also utilized, where the PTT is employed to test whether the Bayes predictor ($\boldsymbol{a}_j$) performs significantly better than a random guessing ($\boldsymbol{a}_{rg}$), denoted as **PTT-Random**.

These paired tests assess the probability (or p-value) of observing a statistically significant difference between the paired samples, i.e., the accuracies of g_p^{mc} and g_p, as described in Section 2.4.1 [46]. The p-value represents the probability of obtaining test results (mean of the accuracy differences) as extreme as the observation, assuming the null hypothesis $H_0(\cdot)$ holds [46]. The null hypothesis $H_0(\boldsymbol{a}_{mc} \approxeq \boldsymbol{a}_j)$ implies that the accuracies are drawn from the same distribution or that the average difference between the paired samples is nearly zero ($\approxeq 0$), suggesting no significant difference in performance, implying no occurrence of IL in the system [46]. The corrected version of PTT, proposed

by [138], is selected among commonly used paired statistical tests, which version accounts for the dependency between the accuracy estimates from conducting a KFCV. However, PTT has limitations, including its reliance on normality assumptions and asymptotic behavior, which can result in overly optimistic p-values. Moreover, accuracy-based measures can be misleading for imbalanced datasets [154, 148], making this approach less reliable. These limitations motivated the usage of FET to analyze the LAS evaluated using MCC (via CM) to detect IL by analyzing the corresponding LAS ($\delta(\mathrm{m}_{\mathrm{MCC}})$).

Fisher's Exact Test Based Approach To address the class imbalance in system datasets and improve p-value estimation accuracy, the FET is proposed for use on the evaluated K CMs (obtained via KFCV) to detect IL, through the LAS evaluated using MCC, i.e., $\delta(\mathrm{m}_{\mathrm{MCC}})$. The p-value tests for independence, with lower values indicating potential IL. The FET provides increased reliability for imbalanced datasets and avoids biases inherent in PTTs. IL is considered likely if a strong correlation exists between inputs $\boldsymbol{x}$ and outputs y in the system dataset $\mathcal{D}$. Since predictions produced by the AutoML pipeline, $\hat{y} = g(\boldsymbol{x})$, are treated as single points encapsulating all input information, the existence of a correlation between $\boldsymbol{x}$ and y implies that $\hat{y}$ contains relevant information for predicting correct outputs (m_{TP}, m_{TN}). Thus, FET is applied to the CM to detect IL in a system by assessing the dependency between predictions $\hat{\boldsymbol{y}}$ and ground truths $\boldsymbol{y}$. This test evaluates the relationships between predicted and actual classes using the CM, making it particularly effective in imbalanced datasets.

FET is a non-parametric test used to determine the probability of independence (or non-dependence) between two classification methods, such as ground truths $\boldsymbol{y}$ and AutoML predictions $\hat{\boldsymbol{y}}$ [64]. It offers the advantage of calculating p-values using the Hypergeometric distribution, ensuring precision even for small datasets. Unlike approximation-based methods that require large sample sizes, FET remains effective in detecting dependencies in small or imbalanced datasets, as discussed in Section 2.4.2. The null hypothesis

$H_0\left(p(\boldsymbol{y}, \hat{\boldsymbol{y}} \mid \boldsymbol{M}) = p(\boldsymbol{y} \mid \boldsymbol{M}) \cdot p(\hat{\boldsymbol{y}} \mid \boldsymbol{M})\right)$ posits that model predictions $\hat{\boldsymbol{y}}$ and ground-truth labels $\boldsymbol{y}$ are independent, suggesting the absence of IL. Conversely, the alternative hypothesis $H_1\left(p(\boldsymbol{y}, \hat{\boldsymbol{y}} \mid \boldsymbol{M}) \neq p(\boldsymbol{y} \mid \boldsymbol{M}) \cdot p(\hat{\boldsymbol{y}} \mid \boldsymbol{M})\right)$ implies significant dependence, indicating the presence of IL.

In the context of ILD, FET is more robust than PTTs, as it directly tests the dependence between classifier predictions $\hat{\boldsymbol{y}}$ and ground-truth labels $\boldsymbol{y}$. Unlike accuracy-based tests that may overestimate performance in imbalanced cases, FET ensures that results are not biased by class imbalance. Notably, the p-value for marginal Bayes predictor using this method is $Pr(\boldsymbol{M}) = 1.0$ because, for predicted class 0 ($\hat{\boldsymbol{y}} = \mathbf{0}$), $\mathrm{m}_{\text{FP}} = 0$, $\mathrm{m}_{\text{TP}} = 0$, and for predicted class 1 ($\hat{\boldsymbol{y}} = \mathbf{1}$), $\mathrm{m}_{\text{FN}} = 0$, $\mathrm{m}_{\text{TN}} = 0$. This approach directly tests the learnability of the empirical risk minimizer g (a proxy for Bayes predictor), independent of marginal Bayes predictor, and relates to MCC, which accounts for class imbalance [24, 32]. Using KFCV, $K = 10$ CMs, are obtained from the j^{th} best-performing pipeline, denoted by $\mathcal{M}_j = \{\boldsymbol{M}_j^k\}_{k=1}^K$, yielding K p-values after applying FET. Bhattacharya and Habtzghi (2002) [17] demonstrate that the *median* of multiple p-values provides the best estimate of the true p-value. Thus, these p-values are aggregated using the *median* and *mean* for robustness, referred to as FET-Median and FET-Mean, respectively. If the p-value is below a predefined significance level (e.g., $\alpha = 0.01$), H_0 is rejected, indicating the presence of IL; otherwise, the test fails to reject the null hypothesis H_0, implying the absence of IL in the system.

Guessing Entropy: Quantification and Detection

In this section, bounds on MI $I(X;Y)$ with respect to GE $G(Y|X)$, using both the McEliece and Yu (1995) [125] upper bound and the Béguinot and Rioul (2024) [11] gamma-parametric lower bound are derived. These bounds relate the LAS in terms of GE $\delta(\mathrm{m}_{\text{GE}}(\boldsymbol{r}^{N_a}, k^*))$ to MI $I(X;Y)$, allowing us to derive conditions based on how $\delta(\mathrm{m}_{\text{GE}}(\boldsymbol{r}^{N_a}, k^*))$ behaves.

McEliece and Yu (1995) [125] Upper Bound By substituting $H(Y|X) = H(Y) - I(X;Y)$ and $G(Y|X) = G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ in the McEliece and Yu (1995) [125] upper bound on GE in terms of conditional entropy in eq. (2.14), the relationship between MI $I(X;Y)$ and the LAS in terms of GE $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ is derived as

$$G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)) \leq \frac{(H(Y) - I(X;Y)) \cdot (M-1)}{2 \cdot \lg(M)} + 1\,,$$
$$I(X;Y) \leq H(Y) - \frac{(G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)) - 1) \cdot 2 \cdot \lg(M)}{M-1}\,.$$

Béguinot and Rioul (2024) [11] Gamma-Parametric Lower Bound Similarly, by substituting $H(Y|X) = H(Y) - I(X;Y)$ and $G(Y|X) = G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ into the gamma-parametric lower bound, which relates the conditional entropy to GE in eq. (2.15), following relationship between MI $I(X;Y)$ and LAS in terms of GE $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ is derived as

$$H(Y) - I(X;Y) = \log_2\left(\gamma \cdot \frac{1-\gamma^M}{1-\gamma}\right) - \log_2(\gamma) \cdot (G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)))\,,$$
$$I(X;Y) = H(Y) - \log_2\left(\gamma \cdot \frac{1-\gamma^M}{1-\gamma}\right) + \log_2(\gamma) \cdot (G(Y) - \delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)))\,.$$

Note that $G(Y) = \frac{(M+1)}{2}$, and when the key bytes are uniformly distributed, $H(Y) = \lg(M)$. As shown in Figure 3.2, in the case where the system leaks maximum information, i.e., $I(X;Y) = H(Y) = \lg(M)$, the LAS $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ reaches its maximum possible value, $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)) = (M-1)/2$. Conversely, when the system leaks no information, i.e., $I(X;Y) = 0$, the LAS $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ is minimized to $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*)) = 0$.

These insights show that LAS in terms of GE $\delta(\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*))$ provides a concrete measure to quantify the IL in an AES-encrypted system, where the secret information is the key byte of the secret key, acting as a direct indicator of the extent of leakage.

Success Rate The GE is approximately inversely proportional to the success rate (SR), i.e., $P_{\text{SR}}(K|Y) \propto \frac{1}{G(Y|X)}$, particularly for small SRs and large numbers of classes ($M \geq 2^8$), as discussed in Section 2.3.3.

A higher SR corresponds to lower GE, reflecting an easier guessing process [11]. The LAS in terms of SR is given by $\delta(\mathrm{m}_{\text{SR}}(\boldsymbol{r}^{N_a}, k^*)) = 1 - \frac{1}{M}$, which implies maximum leakage when $SR = 1$, where $I(X;Y) = \lg(M)$. Conversely, when $\delta(\mathrm{m}_{\text{SR}}(\boldsymbol{r}^{N_a}, k^*)) = \frac{1}{M}$ and $SR = 0$, $I(X;Y) = 0$, indicating no leakage. Although the LAS in terms of SR provides insight into the amount of IL, the GE is chosen as a measure of IL in AES-encrypted systems as it offers more detail regarding the rank of the actual key byte k^*, rather than focusing solely on the first rank. In many systems, multiple guesses g are allowed, typically 3, so for $\delta(\mathrm{m}_{\text{SR}}(\boldsymbol{r}^{N_a}, k^*)) \in \left[\frac{M-1}{2}, \frac{M+1-2g}{2}\right]$, the system is breakable, indicating the presence of IL.

Vulnerability Score and Detection Rather than relying on statistical tests for multiple values of GE, the detection of IL is proposed by calculating the percentage of successful attacks as a direct confidence measure of IL. The VS is derived based on the system's allowed number of guesses and provides a more interpretable measure than binary decisions obtained from statistical tests.

This process is automated using neural architecture search (NAS), streamlining the detection of leakage in AES-encrypted systems through the metric $\mathrm{m}_{\text{VS}}(\boldsymbol{r}^{N_a}, k^*)$, which represents the percentage of *successful attacks*, i.e., when the GE is less than or equal to 3 ($\mathrm{m}_{\text{GE}}(\boldsymbol{r}^{N_a}, k^*) \leq 3$).

As discussed in Section 2.3.3, GE is approximated using R_a different attack datasets $\mathcal{D}_a^{\text{Att}}$ with N_a attack traces, computed as

$$\mathrm{m}_{\text{GE}}(\boldsymbol{r}^{N_a}, k^*) \approxeq \frac{1}{R_a} \sum_{a=1}^{R_a} \boldsymbol{r}^{N_a}[k^*].$$

The percentage of successful attacks, $\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*)$, is estimated by performing A_a different attacks with varying random seeds, as defined in eq. (6.1) as

$$\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*) = \frac{1}{A_a}\sum_{i=1}^{A_a}[\![\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*) \leq 3]\!] \approxeq \frac{1}{A_a}\sum_{i=1}^{A_a}[\![\frac{1}{R_a}\sum_{a=1}^{R_a}\boldsymbol{r}^{N_a}[k^*] \leq 3]\!] .$$

This metric directly measures the AES-encrypted system's vulnerability towards template SCAs. While GE-based approaches are applicable when secret

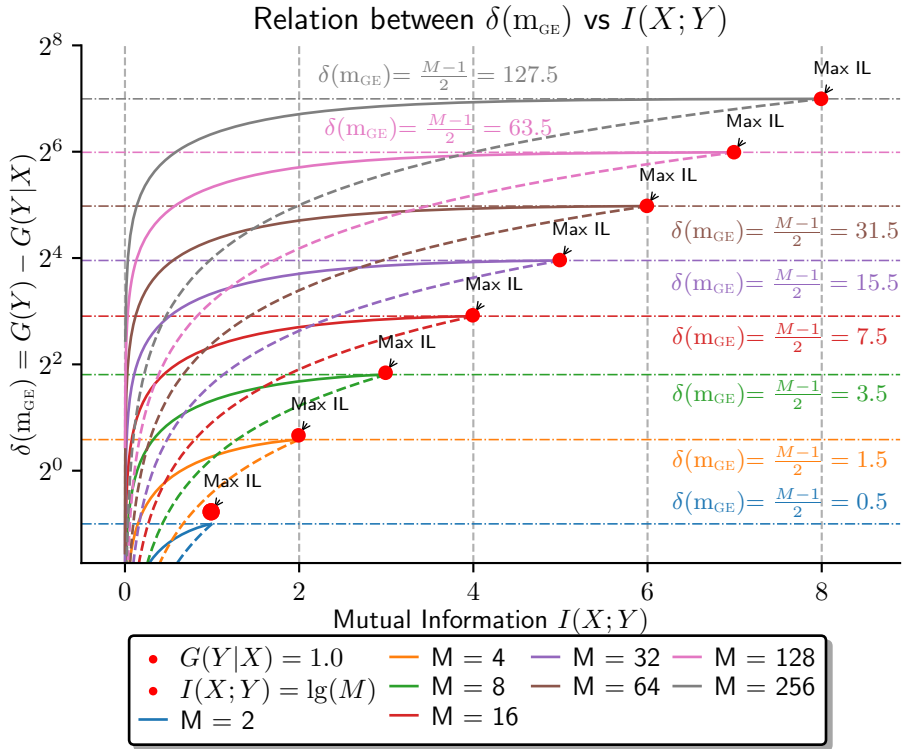


Figure 3.2.: Optimal joint range between the MI ($I(Y|X)$) and GE ($G(Y|X)$) for different classes M in $[2, 256]$ range, with the optimal lower bound derived by McEliece and Yu (1995) [125] shown as dashed line and the upper bound by Béguinot and Rioul (2024) [11] shown as solid line

information corresponds to secret key byte values, the percentage of *successful attacks* ($\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*)$) provides a more interpretable measure of IL in a system. Statistical tests are unnecessary, as VS effectively indicates the probability and extent of IL. Moreover, combined with TST, VS offers insights into SCAs

efficiency and system susceptibility, as detailed in Section 2.3.3. These metrics, explored further in Chapter 6, provide a novel robust framework for assessing vulnerabilities in AES-encrypted systems and evaluating the impact of the template SCAs.

3.3. Mutual Information Estimation Methods

This section introduces two techniques for estimating the MI in a system generating classification datasets by approximating the Bayes predictor, rather than relying solely on standard statistical methods. Mid-Point and Log-Loss are briefly described in Sections 3.3.1 and 3.3.2, respectively. Additionally, Section 3.3.3 outlines three competitive state-of-the-art MI estimation methods [51, 12, 158]. These techniques are used to perform ILD with OTT in a given system, as detailed in Section 3.2.2.

3.3.1. Mid-point Estimation

The Mid-Point approach estimates MI by leveraging the relationship between Bayes error rate $\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right)$ and the conditional entropy $H(Y|X)$ for a classification task [185]. The MI is estimated using the empirical risk minimizer g of eq. (2.5) to approximate the Bayes error rate.

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

$$
\begin{aligned}
&H_l(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right), M) \leq H(Y|X) \leq H_u(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right), M) \\
&H_l(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right), M) = \lg(m) + m(m+1)\left(\lg\left(\frac{m+1}{m}\right)\right)\left(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right) - \frac{m-1}{m}\right) \\
&H_u(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right), M) = H_2(\mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right)) + \mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right)\lg(M-1)\,,
\end{aligned}
$$

where $H_l(\cdot)$ derived by Hellman and Raviv (1970) [85] is valid for $\frac{1}{m+1} \leq 1 - \mathrm{m}_{\mathrm{ER}}\left(g^{bc}\right) \leq \frac{1}{m}\,, m = 1, \ldots, M-1$ and $H_u(\cdot)$ derived by Fano (1961) [56] uses the binary cross-entropy function, $H_2(a) = -a\lg(a) - (1-a)\lg(1-a)$.

Plugging in $H_l(\cdot)$ and $H_u(\cdot)$ in eq. (2.2), the bounds on MI are derived as

$$
\begin{aligned}
&H(Y) - H_u(\mathrm{m}_{\mathrm{ER}}\,(g^{bc}), M) \leq I(X;Y) \leq H(Y) - H_l(\mathrm{m}_{\mathrm{ER}}\,(g^{bc}), M) \\
&F_l(\mathrm{m}_{\mathrm{ER}}\,(g^{bc}), M) \leq I(X;Y) \leq F_u(\mathrm{m}_{\mathrm{ER}}\,(g^{bc}), M) \\
&H(Y) - H_2(\mathrm{m}_{\mathrm{ER}}\,(g^{bc})) \leq I(X;Y) \leq H(Y) - 2\,\mathrm{m}_{\mathrm{ER}}\,(g^{bc})\,, \quad M = 2\,.
\end{aligned}
$$

MI is estimated as

$$
\widehat{I}(X;Y) \approxeq \frac{F_u(\mathrm{m}_{\mathrm{ER}}\,(g), M) + F_l(\mathrm{m}_{\mathrm{ER}}\,(g), M)}{2}\,, \tag{3.1}
$$

where g is the *empirical risk minimizer* of eq. (2.5) and serves as a proxy of the Bayes predictor, as $\mathrm{m}_{\mathrm{ER}}\,(g^{bc}) \approxeq \mathrm{m}_{\mathrm{ER}}\,(g)$.

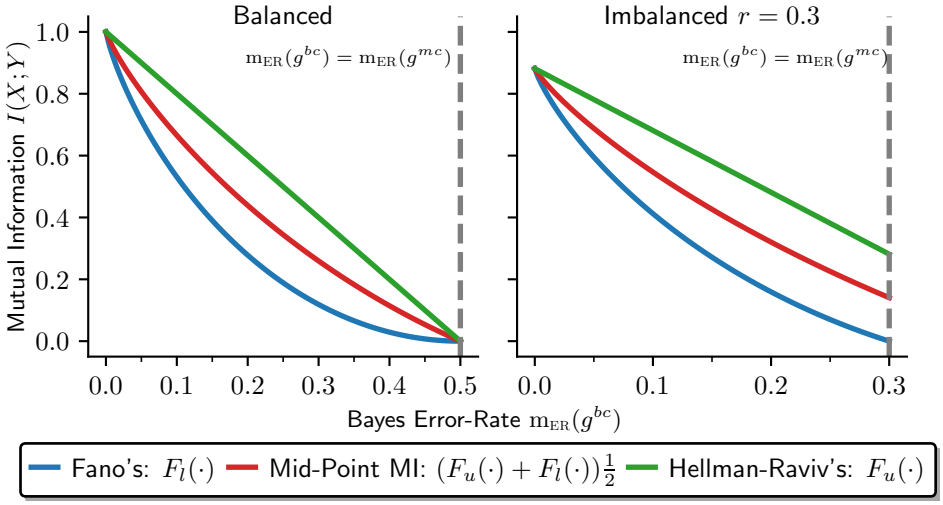


Figure 3.3.: Bayes error rate versus MID-POINT MI (estimated in red)

Limitations in Imbalanced Data As shown in Figure 3.3, for balanced binary classification, when the mid-point MI is 0, the Bayes error rate is 0.5, indicating no IL in the system. However, for the imbalanced case $r = 0.3$, when $\mathrm{m}_{\mathrm{ER}}\,(g^{bc}) = 0.3$, the mid-point MI is more than 0, despite no actual IL

according to the statistical learning theory condition in Section 3.2.1. This highlights the issue of overestimation in the Mid-Point approach, where the true MI might correspond to the lower bound $F_l(\mathrm{m}_{\mathrm{ER}}(g^{bc}), M)$.

To illustrate this issue, let us consider an example where the class imbalanced joint distribution $p_{(X,Y)}(\cdot)$, along with the calculated marginals $p_Y(\cdot)$, $p_X(\cdot)$, and the conditional $p_{Y|X}(\cdot)$, is defined as

Joint Distribution			
$p_{(X,Y)}(x,y)$	$y=0$	$y=1$	$p_X(x)$
$x=0$	0.5	0	0.5
$x=1$	0.45	0.05	0.5
$p_Y(y)$	0.95	0.05	–

Conditionals on Y given X			
$p_{Y\|X}(y\,\|\,x)$	$y=0$	$y=1$	$p_X(x)$
$x=0$	1	0	0.5
$x=1$	0.9	0.1	0.5
$p_Y(y)$	0.95	0.05	–

(3.2)

In this scenario, the ground-truth MI is calculated by substituting the joint and marginal probability mass functions (PMFs) from eq. (2.2) into eq. (2.3) as $I(X;Y) = \sum_{x\in\{0,1\}} \sum_{y\in\{0,1\}} p_{(X,Y)}(x,y) \lg\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right) = 0.0519\,\mathsf{bits}$. The optimal Bayes predictor for decision-making is defined as $g^{bc}(x) = 0$ using eq. (2.6), yielding a Bayes error rate of 0.05 ($\mathrm{m}_{\mathrm{ER}}(g^{bc}) = \sum_{x\in\{0,1\}} p_X(x)\cdot p_{Y|X}(y=1\,|\,x) = 0.05$). The binary entropy of Bayes error rate and Y is then calculated using eq. (2.1), resulting in $H(Y) = H_2(\mathrm{m}_{\mathrm{ER}}(g^{bc})) \approx 0.286\,\mathsf{bits}$. By inserting these values into eq. (3.1), the Mid-Point MI estimate is approximately as $0.0932\,\mathsf{bits}$.

In this example, the ground-truth MI is $0.0519\,\mathsf{bits}$, while the Mid-Point MI estimate is $0.0932\,\mathsf{bits}$, demonstrating the issue of overestimation. This discrepancy arises from the deterministic nature of error rates in capturing the probabilistic relationship (MI) between the variables. error rates, therefore, may not always be an accurate metric for assessing performance in imbalanced datasets, potentially detecting non-existent ILs in the system, leading to false positives.

To improve the estimation accuracy using these bounds, it is essential to define a Bayes predictor that minimizes metrics such as balanced error-rate (BER) or maximizes other metrics like MCC or F1-score, as described in Section 2.2.2. [210] establishes a relationship between BER and MI for binary classification; however, this relationship does not generalize to multi-class scenarios.

Baseline ILDs Oversights One common approach to identifying IL is the PTT-Majority baseline, which relies on uncovering significant differences in error rates between the Bayes predictor (or a suitable approximation g or g_p) and the marginal Bayes predictor [80]. However, considering the example in eq. (3.2), the Bayes predictor ($g^{bc}(x) = 0$) coincides with marginal Bayes predictor and always predicts class 0 for any given input x. This alignment between Bayes predictor and marginal Bayes predictor leads to a situation where the IL present in a system goes undetected by PTT-Majority. Furthermore, the FET based ILD approaches, including FET-Mean and FET-Median, also fall short in identifying this IL. The failure to identify the IL stems from the Bayes predictor's CM, which in this case is evaluated as $\left(\begin{smallmatrix} 95 & 00 \\ 05 & 00 \end{smallmatrix}\right)$, yielding an exact p-value of 1 when tested using the FET [80]. Consequently, in cases when Bayes error rate is very close to marginal Bayes predictor's error rate, these baseline ILD approaches fail to detect the IL in the system and produce false negatives [80].

3.3.2. Log-Loss Estimation

The Log-Loss approach for MI estimation is proposed to address the overestimation and false positives of the Mid-Point approach, as well as the false negatives of baseline ILD approaches. This approach employs the empirical risk minimizer g_p, which minimizes eq. (2.8), to approximate the log-loss of the Bayes predictor.

The MI between X and Y in eq. (2.2) is defined as

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

where $\mathbb{E}[\ell_{ll}(\cdot)]$ represents the expected log-loss. The conditional entropy $H(Y|X)$ equals the expected log-loss of the Bayes predictor g^{bc}, and the entropy of Y is the expected log-loss of the marginal Bayes predictor g^{mc}, making it a special case of LAS, as discussed in Section 3.2.1.

Log-loss For probabilistic classifiers g_p, categorical cross-entropy (CCE) is used to obtain estimated conditional class probabilities, forming a vector $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_M)$ for each input $\boldsymbol{x}$. The log-loss for g_p is defined in **[** (**[**) [**[**]chap. 4]bishop2006pattern as

$$\ell_{ll}(y, \hat{\boldsymbol{p}}) = -\sum_{m=1}^{M} \hat{p}_m \lg(\hat{p}_m) = -\sum_{m=1}^{M} g_p(\boldsymbol{x})[m] \lg(g_p(\boldsymbol{x})[m])\,.$$

It reaches its minimum value of 0 when the class probability is high for one class m, i.e., $\hat{p}_m \approxeq 1$ and low for remaining, i.e., $\hat{p}_j \approxeq 0\,, \forall j \in [M] \setminus \{m\}$, and its maximum value $\lg(M)$ when probabilities are uniformly distributed, i.e., $\hat{p}_m = {}^1\!/_M\,, \forall m \in [M]$, implying that high uncertainty in the classification also increases the log-loss, making it suitable to estimate the conditional entropy $H(Y|X)$.

Marginal Bayes predictor The marginal Bayes predictor, denoted by g_p^{mc}, is estimated from the class distribution in the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ using eq. (2.8). The class probabilities for each input $\boldsymbol{x}$ are

$$g_p^{mc}(\boldsymbol{x}) = \hat{\boldsymbol{p}}_{mc} = (\hat{p}_1, \ldots, \hat{p}_M)\,, \tag{3.3}$$

where $\hat{p}_m = \frac{|\{(\boldsymbol{x}_i, y_i) \in \mathcal{D} \mid y_i = m\}|}{|\mathcal{D}|}$ is the fraction of instances for class m. The log-loss of the marginal Bayes predictor suitably estimates entropy of Y ($H(Y)$), i.e., $H(Y) \approxeq \ell_{ll}(y, \hat{\boldsymbol{p}}_{mc})$, ranging from 0 and $\lg(M)$, attaining the maximum value for a balanced dataset.

MI estimation The expected log-loss of the Bayes predictor and marginal Bayes predictor is approximated 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

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

The MI is then estimated as

$$\widehat{I}(X;Y) \approxeq \sum_{m=1}^{M}\left(\frac{1}{N}\sum_{i=1}^{N} g_p(\boldsymbol{x}_i)[m] \lg(g_p(\boldsymbol{x}_i)[m]) - \hat{\boldsymbol{p}}_{mc}[m] \lg(\hat{\boldsymbol{p}}_{mc}[m])\right). \quad (3.4)$$

Classifier Calibration To enhance Log-Loss estimation accuracy, five prevalent multi-class calibration approaches are utilized, namely Platt's Scaling (**PS Cal Log-Loss**), Isotonic Regression (**IR Cal Log-Loss**), Beta Calibration (**Beta Cal Log-Loss**), Temperature Scaling (**TS Cal Log-Loss**), and Histogram Binning (**HB Cal Log-Loss**). These methods, collectively referred to as Cal Log-Loss, are described in Section 2.2.3. These techniques are applied to improve the precision of Log-Loss, estimating MI as 0.052 ***bits*** exactly for eq. (3.2), thereby avoiding false positives as observed with the Mid-Point approach. When these techniques are used to estimate MI and perform ILD via Log-Loss, they are collectively denoted as **Cal Log-Loss**.

3.3.3. Baselines

This section will briefly describe three baseline approaches recently introduced to estimate the MI, namely Gaussian mixture model (GMM), mutual information neural estimation (MINE) and PC-Softmax [51, 12, 158].

Gaussian Mixture Model

GMMs are widely used to estimate MI in classification datasets $\mathcal{D}$ by approximating unknown probability density functions (PDFs) such as joint, marginal, and conditional distributions ($p_{(X,Y)}$, p_X, p_Y) [121].

Estimation Process The estimation process begins by fitting a GMM to the joint space $\mathcal{X} \times \mathcal{Y}$ to obtain the joint PDF $\hat{p}_{(X,Y)}(\cdot)$, which is then used to derive the marginal PDFs on X ($\hat{p}_X(\cdot)$) and Y ($\hat{p}_Y(\cdot)$). The MI is then computed using the estimated PDFs ($\hat{p}(\cdot)$) and the given dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ using eq. (2.3) as

$$\widehat{I}(X;Y) = \frac{1}{N} \sum_{i=1}^{N} \left(\lg(\hat{p}_{(X,Y)}(X = \boldsymbol{x}_i, Y = y_i)) - \lg(\hat{p}_X(\boldsymbol{x}_i)) - \lg(\hat{p}_Y(y_i)) \right) .$$

In a GMM with K components, each component k has unique parameters: mixing coefficients (π_k), means (μ_k), and covariance matrices (Σ_k), such that the sum of all coefficients must equal one, i.e., $\sum_{k=1}^{K} \pi_k = 1$ [51]. Each mixing coefficient π_k is between 0 and 1, i.e., $0 < \pi_k < 1$ and is expressed as

$$\mu_k = \begin{bmatrix} \mu_{X_k} \\ \mu_{Y_k} \end{bmatrix}, \quad \Sigma_k = \begin{bmatrix} \Sigma_{XX_k} & \Sigma_{XY_k} \\ \Sigma_{YX_k} & \Sigma_{YY_k} \end{bmatrix} .$$

The marginal GMM for X ($\hat{p}_X(\cdot)$) is constructed using the same K components, with mixing coefficients (π_k), means (μ_{X_k}) and covariance matrices (Σ_{XX_k}) [51]. Similarly, the marginal GMM for Y ($\hat{p}_Y(\cdot)$) uses the corresponding mixing

coefficients (π_k), means (μ_{Y_k}) and covariance matrices (Σ_{YY_k}). The expectation-maximization (EM) algorithm is used to fit the GMM and determine the optimal number of components K since too few components may miss key data patterns, while too many can cause overfitting.

Curse of Dimensionality Although GMMs are robust and flexible for modeling complex data distributions, they suffer from the curse of dimensionality due to the quadratic growth of parameters ($O(d^2)$), making them prone to overfitting and computational inefficiency in high-dimensional settings [121, 51]. To overcome these issues,high-dimensional data clustering (HDDC) reduces parameters by imposing constraints on covariance matrices, focusing on the most informative dimensions (largest eigenvalues) and simplifying others [51].

To avoid overfitting in high-dimensional data, HDDC uses diagonal covariance (assuming feature independence), spherical covariance (assumed equal variance in all directions) suitable for isotropic data distributions, and tied covariance (shared structure across components) to minimize the number of free parameters [121]. While these strategies address the curse of dimensionality, they still struggle with complex distributions and strong dependencies despite ensuring compatibility with the EM algorithm [51, 121].

Improvements To address these limitations, the GMM framework in the `InfoSelect` library is extended by incorporating diagonal, spherical, and tied covariance structures to balance flexibility and computational efficiency[1] [153]. Diagonal and spherical constraints effectively reduce free parameters in high-dimensional data, minimizing overfitting, while tied covariance matrices provide stability and robustness, especially when clusters share similar patterns. Dimensionality reduction is applied for datasets with more than 100 features, preserving the most informative variables to improve MI estimation. For robustness and fair comparison, Akaike information criterion (AIC) is used as the objective function for hyperparameter optimization (HPO) to select the

[1] `https://github.com/felipemaiapolo/infoselect/issues/5`

optimal model and prevent overfitting. The AIC assists in this decision by considering both the log-likelihood $\ell(\eta)$ of the fitted model and the number of free parameters F, defined as $\text{AIC} = -2\log\ell(\eta) + 2F$ [51]. Lower AIC values reflect a better fit by balancing likelihood and complexity, ensuring that the GMM captures essential patterns without overfitting.

Mutual Information Neural Estimation

MINE estimates the MI between two variables $(X, , Y)$ by optimizing the Donsker-Varadhan (DV) representation, providing a lower-bound estimate of MI, which makes it suitable for high-dimensional data [12]. However, the DV bound can introduce numerical instability, necessitating techniques such as gradient clipping and careful learning rate scheduling to maintain stability during training.MINE is particularly useful for representation learning, feature selection, and measuring dependencies in latent variables for deep learning (DL) models.

Estimation Process The MINE loss function used to train a neural network $T_\theta(\boldsymbol{x}, y) : \mathbb{R}^{d+M} \to \mathbb{R}$ with learnable parameters θ using the dataset $\mathcal{D} = \{\boldsymbol{x}i, y\}i = 1^N$ is defined a

$$\ell_{\text{MINE}}(\boldsymbol{x}, y) = \sup_\theta \left(\mathbb{E}_{(\boldsymbol{x},y)\sim p_{(X,Y)}(\boldsymbol{x},y)}[T_\theta(\boldsymbol{x}, y)] - \log(\mathbb{E}_{(\boldsymbol{x},y)\sim p_X(\boldsymbol{x})p_Y(y)}[\mathbf{e}^{(T_\theta(\boldsymbol{x},y))}]) \right) ,$$

The d-dimensional input $\boldsymbol{x}$ concatenated with one-hot-encoded M length ground-truth class label y serves as an input to the network. The network architecture includes an input layer (`Linear` units), hidden layers with `ReLU` activation, and a real-valued output layer (`Linear` unit) to quantify the dependency between $\boldsymbol{x}$ and y. MINE works by maximizing the discrepancy between the joint distribution $(p_{(X,Y)}(\cdot))$ and the product of marginals $(p_X(\cdot)p_Y(\cdot))$, allowing it to capture dependencies between variables without explicitly estimating their

underlying distributions [12]. MINE estimates MI through iterative training process outlined in Algorithm 1, making it effective for high-dimensional data with complex, unobserved PDFs.

Algorithm 1 Training algorithm: MINE

Require: Dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, Neural Network T_θ
Ensure: Estimated MI: $\hat{I}_\theta$
Initialize network parameters θ and hyperparameters
Set $N_{epochs} = 10000$, Set B such that $N \mod B = 0$
for $n \in [N_{epochs}]$ **do**
    Define a random permutation π on set of indices $[N]$
    for $b \in [B]$ **do**
        Define batch start and end: $b_s = (b-1)\frac{N}{B} + 1$, $b_e = b \cdot \frac{N}{B}$
        Evaluate the lower-bound:
        $V(\theta) \leftarrow \frac{1}{B}\sum_{i=b_s}^{b_e} T_\theta(\boldsymbol{x}_i, y_i) - \log\left(\frac{1}{B}\sum_{i=b_s}^{b_e}\left(\exp\left(T_\theta(\boldsymbol{x}_i, y_{\pi(i)})\right)\right)\right)$
        Evaluate bias-corrected gradients (using exponential moving average) as
        $G_b(\theta) \leftarrow \nabla_\theta V(\theta)$
        Update the statistics network parameters: $\theta \leftarrow \theta + G_b(\theta)$
    end for
end for
Compute $\hat{I}_\theta = \frac{1}{N}\sum_{i=1}^{N} T_\theta(\boldsymbol{x}_i, y_i) - \log\left(\frac{1}{N}\sum_{i=1}^{N}\left(\exp\left(T_\theta(\boldsymbol{x}_i, y_{\pi(i)})\right)\right)\right)$
return Estimated MI: $\lg(\mathrm{e}) \cdot \hat{I}_\theta$

Improvements The main challenge for training MINE using Algorithm 1 lies in determining the suitable neural network architecture for estimating MI since a well-defined objective function is needed to identify the optimal neural network architecture for accurate MI estimation using standard HPO methods, leading to potential overfitting and biased results. Discrepancy-based methods like MINE and Kullback-Leibler (KL) divergence, which compare network outputs between shuffled and original datasets, can potentially lead to architectures that overfit and produce biased MI estimates. An ensemble approach addresses this by combining the outputs of the top-5 models identified through HPO

with mean squared error (MSE) as the objective function. This strategy improves estimation precision and robustness, ensuring reliable performance across diverse datasets.

PC-Softmax

PC-Softmax provides an alternative approach to estimating MI in classification datasets $\mathcal{D}$ using a modified *softmax* function, known as probabilitiy-corrected softmax (PC-softmax) [158]. This method integrates PC-softmax with the CCE loss function to derive an empirical risk minimizer g_p minimizing eq. (2.8) for the given classification datasets $\mathcal{D}$. The (conditional) class probabilities predicted by g_p are subsequently used to estimate MI through the PC-softmax function [158].

One of the main advantages of this approach over methods like MINE is its simplicity and the ease with which the network architecture can be optimized using classification evaluation metrics or loss functions as objective functions. PC-softmax approximates a lower bound on MI by estimating the KL divergence between the joint distribution $p_{(X,Y)}(\cdot)$ and the product of marginals $p_X(\cdot), p_Y(\cdot)$ for variables X and Y [158].

Estimation Process Estimating (conditional) class probabilities by the empirical risk minimizer g_p is challenging compared to standard classification tasks. Typically, g_p is obtained by learning a scoring function $h_{s|\boldsymbol{\theta}}(\boldsymbol{x})$, which assigns un-normalized real-valued scores to each class label, parameterized by $\boldsymbol{\theta}$. Neural networks or MLPs are commonly used for this purpose, where $h_{s|\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{s}$ outputs a vector of un-normalized scores $\boldsymbol{s} = (s_1, \ldots, s_M)$ for a given instance $\boldsymbol{x}$, where $s_m = h_{s|\boldsymbol{\theta}}(\boldsymbol{x})[m] = \boldsymbol{s}[m]$ is the score assigned to class m. The *softmax* function $S(\boldsymbol{x})\colon \mathbb{R}^M \to [0,1]^M$ is then commonly used to convert these scores into probabilities $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_M)$, typically implemented in the output layer with M nodes of the MLP. This function transforms the score

$\boldsymbol{s}[y]$ to the corresponding (conditional) class probability $\hat{\boldsymbol{p}}[y]$ as for a given ground-truth class $y \in [M]$ as

$$S(\boldsymbol{x})[y] = \hat{\boldsymbol{p}}[y] = \frac{\exp(\boldsymbol{s}[y])}{\sum_{m=1}^{M} \exp(\boldsymbol{s}[m])} .$$

The empirical risk minimizer g_p is then obtained by minimizing the CCE loss, defined as $\ell_{CCE}(\boldsymbol{x}, y) = -\ln(S(\boldsymbol{x})[y])$.

Qin and Kim (2019) [158] demonstrates that the expected CCE loss approximates MI between the input $\mathcal{X}$ and output $\mathcal{Y}$, up to a constant $\lg(M)$ under a uniform label distribution, i.e., $I(X;Y) \geq \widehat{I}(X;Y) \approxeq -\sum_{(\boldsymbol{x},y)\in\mathcal{D}} \log(S(\boldsymbol{x})[y])$. To further improve MI estimation, the authors extend the traditional softmax to PC-softmax, defined as

$$S_{pc}(\boldsymbol{x})[y] = \frac{\exp(\boldsymbol{s}[y])}{\sum_{m=1}^{M} \exp(p_Y(m) \cdot \boldsymbol{s}[m])} .$$

The PC-softmax reduces to the traditional *softmax* when $p_Y(y) = \frac{1}{M}$ for all y. The corresponding CCE loss is $\ell_{\text{PCCE}}(\boldsymbol{x}, y) = -\ln(S_{pc}(\boldsymbol{x})[y])$, which improves MI estimation, especially for imbalanced datasets. To compute this, the marginal distribution $p_Y(\cdot)$ is typically estimated from the dataset as $\hat{p}_Y(y) = \frac{|\{(\boldsymbol{x}_i,y_i)\in\mathcal{D};|;y_i=y\}|}{|\mathcal{D}|}$. This loss function maximizes the lower bound on the MI, making it a simple yet effective technique for estimating MI in classification tasks. The MI for a given dataset $\mathcal{D} = \{\boldsymbol{x}_i, y\}_{i=1}^{N}$ is estimated as

$$I(X;Y) \geq \widehat{I}(X;Y) \approxeq \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \ell_{\text{PCCE}}(\boldsymbol{x}, y) \times \lg(\mathrm{e}) = -\frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \lg(S_{pc}(\boldsymbol{x})[y]) .$$

This approach provides a *lower bound* on MI but offers a simpler and more flexible method for estimation. In contrast to MINE, the MLP architecture used here can be optimized for any given dataset $\mathcal{D}$ using standard objective functions like accuracy or validation CCE loss.

4. Mutual Information Estimation

This chapter evaluates methods for estimating mutual information (MI), comparing them to state-of-the-art approaches using datasets generated by synthetic systems (*synthetic datasets*). The terms *systems* and datasets are used interchangeably, as MI is estimated through these systems—a key step for detecting information leakage (IL) via MI-based information leakage detection (ILD) approaches, as discussed in Section 3.2.2. The chapter begins with the generation of synthetic datasets using the multivariate normal (MVN) distribution, detailed in Section 4.1. Section 4.2 outlines the empirical setup, employing the normalized mean absolute error (NMAE) metric for performance evaluation. Results are presented in Section 4.3.2, with generalization capabilities analyzed across factors such as number of classes (M), input dimensions (d), class imbalance (r), and noise levels (ϵ), summarized in Section 4.3.2 and detailed in Appendix A.4. The findings demonstrate the superior performance and generalization of the proposed approaches over state-of-the-art methods. This chapter is based on the work published in Gupta et al. (2024) [81].

4.1. Simulating Synthetic Systems

The MVN distribution is employed for simulating real-world vulnerable and non-vulnerable systems that generate classification datasets, providing a straightforward means of obtaining ground truth MI, as described in Section 4.1.3.

The MVN distribution is a natural choice due to its simplicity, well-established statistical properties, and ability to model complex inter-variable correlations in real-world data effectively, and it is widely employed for benchmarking classifiers [19, chap. 2]. The central limit theorem further justifies this choice, stating that the cumulative sum of many independent random variables always conforms to a normal distribution [19, chap. 4]. The systems are also simulated to generate both balanced and imbalanced datasets with varying leakage assessment score (LAS) or MI levels using the MVN perturbation and proximity techniques. Some illustrations of two-dimensional data points generated using the two noise introduction procedures are shown in Figure 4.2.

4.1.1. Generation Method

Synthetic datasets are generated using MVN to define the joint probability density function (PDF) ($p_{(X,Y)}$) between X and Y, inducing their marginals on X and Y. The dataset $\mathcal{D}$ is created by sampling instances from $p_{X|Y}(\cdot)$ with class distribution $p_Y(y)$, as illustrated in Figure 4.2.

Formal Definition of PDFs

The joint distribution $p_{(X,Y)}(\cdot)$ and marginal on Y $p_X(\cdot)$ is defined which is required to generate the dataset $\mathcal{D}$ by sampling $(\boldsymbol{x}_i, y_i) \sim p_{X|Y}(\boldsymbol{x}_i \,|\, y_i), \forall i \in [N]$ 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 m as

$$p_{X|Y}(\boldsymbol{x} \,|\, m) = \mathrm{MVN}(\boldsymbol{\mu}_m, \Sigma)\,. \tag{4.1}$$

where Σ is the covariance matrix and $\boldsymbol{\mu}_m$ is the mean vector for class m. The covariance matrix should be positive semi-definite and is typically sampled using eigenvalue decomposition [19], defined as

$$\begin{aligned}
&\boldsymbol{Q} \sim \text{OrthogonalMatrix}_{d\times d} = \begin{pmatrix} q_{1,1} & \cdots & q_{1,d} \\ \cdots & \cdots & \cdots \\ q_{d,1} & \cdots & q_{d,d} \end{pmatrix},\ q_{i,j} \in [-1,1),\ \forall (i,j) \in [d] \times [d] \\
&\boldsymbol{S} \sim \text{Diagonal}(\text{RandomMatrix}_{d\times d}) = \begin{pmatrix} s_{1,1} & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & s_{d,d} \end{pmatrix},\ s_{i,i} \in [0,1),\ \forall i \in [d] \\
&\Sigma = (\boldsymbol{Q} \cdot \boldsymbol{S}) \cdot \boldsymbol{Q}^T. \qquad (4.2)
\end{aligned}$$

Introducing Class Imbalance (r): Marginal on Y The parameter r is the minimum proportion of instances for any class $m \in [M]$, defined as $r = \min_{m\in[M]} \frac{|\{(\boldsymbol{x}_i,y_i)\in\mathcal{D} \mid y_i=m\}|}{|\mathcal{D}|}$. For balanced datasets, $r = 1/M$, and for imbalanced ones, $r < 1/M$, i.e., $r \in (0, 1/M]$. To generate multi-class imbalanced datasets ($M > 2$), the two methods are proposed: *Minority* and *Majority*, with an example class frequency in each case shown in Figure 4.1. In the *Minority* method, the **minority class** is assigned r fraction of total data points, and remaining samples are uniformly distributed among other classes ($> \frac{N}{M}$). While in the *Majority* method, all classes apart from the selected **majority class** is assigned r fraction of total data points ($< \frac{N}{M}$), and the **majority class** gets the remaining data points, To generate imbalanced datasets using MVN, the vector $\boldsymbol{n}_M^{\mathbf{g}_r} = (n_1, \ldots, n_M)$ is defined, where n_m denotes the sample counts for each class $m \in [M]$, defined for each generation method as

$$\begin{aligned}
\boldsymbol{n}_M^{\text{Majority}} &= (\lceil N \cdot r \rceil, \ldots, \lceil N \cdot r \rceil, N - (M-1)\lceil N \cdot r \rceil) \\
\boldsymbol{n}_M^{\text{Minority}} &= \left(\lceil \frac{N - \lceil N \cdot r \rceil}{(M-1)} \rceil, \ldots, \lceil \frac{N - \lceil N \cdot r \rceil}{(M-1)} \rceil, \lceil N \cdot r \rceil \right). \qquad (4.3)
\end{aligned}$$

The class frequencies for each generation method are depicted in Figure 4.1.

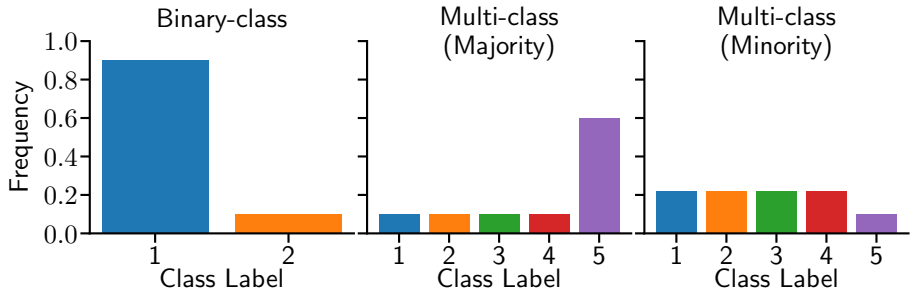


Figure 4.1.: Class frequencies in generated imbalanced datasets

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

$$\mathsf{g}_r = \begin{cases} \text{Majority} & \begin{cases} p_Y(m) = r, & \text{if } m \in [M]_0 \setminus M \\ p_Y(m) = 1 - r \cdot (M-1), & \text{if } m = M \end{cases} \\ \text{Minority} & \begin{cases} p_Y(m) = \frac{1-r}{M-1}, & \text{if } m \in [M]_0 \setminus M \\ p_Y(m) = r, & \text{if } m = M \end{cases} \end{cases} \tag{4.4}$$

Joint PDF between X and Y Using these, the joint distribution $p_{(X,Y)}(\boldsymbol{x}, m)$ for class m is defined as

$$p_{(X,Y)}(\boldsymbol{x}, m) = p_Y(m) \cdot p_{X|Y}(\boldsymbol{x} \,|\, m) \tag{4.5}$$

Marginal on X The joint probability in eq. (4.5) induced the marginal on X as

$$p_X(\boldsymbol{x}) = \sum_{m=1}^{M} p_Y(m) \cdot p_{X|Y}(\boldsymbol{x} \,|\, m) = \sum_{m=1}^{M} p_Y(m) \cdot \mathrm{MVN}(\mu_m, \Sigma) \tag{4.6}$$

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

$$p_{Y|X}(m \mid \boldsymbol{x}) = \frac{p_Y(m) \cdot p_{X|Y}(\boldsymbol{x} \mid m)}{\sum_{m=1}^{M} p_Y(m) \cdot p_{X|Y}(\boldsymbol{x} \mid m)} = \frac{p_{(X,Y)}(\boldsymbol{x}, m)}{p_X(\boldsymbol{x})}. \tag{4.7}$$

4.1.2. Introducing Noise (ϵ) in the System

To simulate real-world IL scenarios in cryptographic systems, the noise is introduced to decrease 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 $I(X;Y) < H(Y)$. The two techniques are proposed called 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 ($_{\mu}m$) of each class, leading to overlap between the Gaussian clusters ($\text{MVN}(\boldsymbol{\mu}_m, \Sigma), \forall m \in [M]_0$), outlined in Algorithms 2 and 3, respectively. These methods simulate scenarios where cryptographic systems leak no information (non-vulnerable) with $I(X;Y) = 0$.

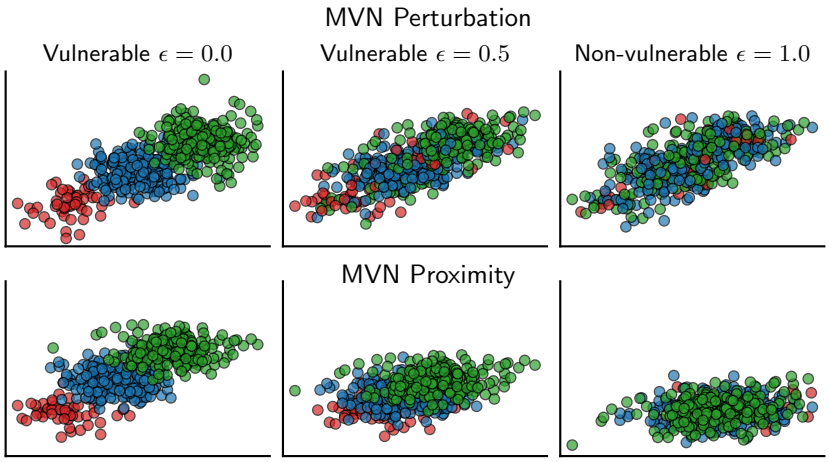


Figure 4.2.: MVN perturbation and proximity techniques: synthetic datasets

MVN Perturbation Technique

This approach introduces noise by flipping a percentage of class labels in the dataset, simulating perturbed systems to generate classification datasets with specified configurations, including the generation method (g_r), number of classes (M), input dimensions (d), noise (ϵ), and class imbalance (r), as outlined in Algorithm 2. Flipping ϵ percentage of class labels ($y \sim \mathcal{Y}$) modifies the original conditional PDFs $p_{Y|X}(\cdot)$ and $p_{X|Y}(\cdot)$, defined in eqs. (4.1) and (4.7), respectively.

Modified Output Variable Y_ϵ Let random variables $B \sim \text{Bernoulli}(p = \epsilon, q = 1 - \epsilon)$ and $\tilde{Y} \sim \text{Categorical}(p_Y(1), \ldots, p_Y(M))$ 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.

Algorithm 2 Generate MVN Perturbation Synthetic Dataset $\mathcal{D}$

Given g_r, M, d, ϵ and r
Define $\mathcal{D} = \{\}$, $N = 1000 \cdot M$
Calculate $\boldsymbol{n}_M^{\mathsf{g}_r} = (n_1, \ldots, n_M)$ using eq. (4.3)
Sample the positive semi-definite Σ of MVNs using eq. (4.2)
for $m \in [M]$ **do**
    Define $\boldsymbol{\mu_m} = (1.5m, \ldots, 1.5m) \in \mathbb{R}^d$
    for $i \in [n_m]$ **do**
        Sample $\boldsymbol{x}_i \sim \text{MVN}(\boldsymbol{\mu_m}, \Sigma)$
        Assign label y_i using $B \sim \text{Bernoulli}(p = \epsilon, q = 1 - \epsilon)$ ▷ Flip ϵ % labels

$$\begin{cases} y_i = m, & B = 0 \\ y_i \sim \text{Categorical}(p_Y(1), \ldots, p_Y(M)), & B = 1 \end{cases}$$

        $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{x}_i, y_i)\}$ ▷ Add instances
    end for
end for
return $\mathcal{D}$

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

$$\begin{aligned} p_{Y_\epsilon|X}(Y_\epsilon = m \,|\, \boldsymbol{x}) &= p_B(B=0) \cdot p_{Y|X}(m \,|\, \boldsymbol{x}) + p_B(B=1) \cdot p_{\tilde{Y}}(\tilde{Y} = m) \\ &= (1-\epsilon) \cdot p_{Y|X}(m \,|\, \boldsymbol{x}) + \epsilon \cdot p_Y(m)\,. \end{aligned} \tag{4.8}$$

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

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

The altered conditional distribution $p_{X|Y_\epsilon}(\boldsymbol{x} \,|\, Y_\epsilon = m)$ becomes a combination of multiple MVN distributions, increasing complexity and making MI estimation more challenging.

Algorithm 3 Generate MVN Proximity Synthetic Dataset $\mathcal{D}$

1: Given g_r, M, d, ϵ and r
2: Define $\mathcal{D} = \{\}$, $N = 1000 \cdot M$
3: Calculate $\boldsymbol{n}_M^{\mathsf{g}_r} = (n_1, \ldots, n_M)$ using eq. (4.3)
4: Sample the positive semi-definite Σ of MVNsusing eq. (4.2)
5: **for** $m \in [M]$ **do**
6: Define $\boldsymbol{\mu_m} = (1.5 \cdot m(1-\epsilon), \ldots, 1.5 \cdot m(1-\epsilon)) \in \mathbb{R}^d$ ▷ Distance $1.5(1-\epsilon)$
7: **for** $i \in [n_m]$ **do**
8: Sample $\boldsymbol{x}_i \sim \mathsf{MVN}(\boldsymbol{\mu_m}, \Sigma)$
9: $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{x}_i, m)\}$ ▷ Add instances
10: **end for**
11: **end for**
12: **return** $\mathcal{D}$

MVN Proximity Technique

The 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 $\boldsymbol{\mu}'_m, \forall m \in [M]_0$ corresponding to MVN distribution representing each class closer to each other. The updated MVNs for each class m is defined by $\mathrm{MVN}(\boldsymbol{\mu}'_m, \Sigma)$, which in turn represents the conditional PDF $p_{X|Y}(\cdot)$ for the underlying generated system dataset. The process for simulating systems using the proximity technique to generate classification datasets for a unique configuration of the generation method (g_r), number of classes (M), input dimensions (d), noise (ϵ), and class imbalance (r), is outlined in Algorithm 3. Notably, when introducing proximity, only the original MVN is modified, which only modifies the conditional on X given Y, i.e., $p'_{X|Y}(\boldsymbol{x} \,|\, m) = \mathrm{MVN}(\boldsymbol{\mu}'_m, \Sigma)$. Consequently, the conditional probability $p_{Y|X}(\cdot)$ and the marginal on X can be computed using eqs. (4.6) and (4.7) as

$$p'_{Y|X}(m \,|\, \boldsymbol{x}) = \frac{p_Y(m) \cdot p'_{X|Y}(\boldsymbol{x} \,|\, m)}{\sum_{m=1}^{M} p_Y(m) \cdot p'_{X|Y}(\boldsymbol{x} \,|\, m)} = \frac{p'_{(X,Y)}(\boldsymbol{x}, m)}{p_X(\boldsymbol{x})}\,. \tag{4.9}$$

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.

4.1.3. Ground-truth MI

To evaluate the MI estimation methods, it is essential to calculate the ground-truth MI for synthetic datasets generated using the conditional PDF $p_{X|Y}(\cdot)$ and the marginal $p_Y(\cdot)$ on Y, as defined in eqs. (4.1) and (4.4), respectively. The maximum MI, i.e., the entropy of Y, is obtained by substituting $p_Y(\cdot)$ into eq. (2.1). For balanced datasets ($r = \frac{1}{M}$), the entropy of Y equals $\lg(M)$, while for imbalanced datasets ($r \in (0, \frac{1}{M})$), the entropy is lower than $\lg(M)$. By plugging in the conditional $p_{Y|X}(\cdot)$ and the marginal $p_Y(\cdot)$ defined above

in eq. (2.2), the ground truth MI for the generated system dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}$ is approximated as

$$GI(\mathcal{D}) \approxeq \frac{1}{N} \sum_{i=1}^{N} p_{Y|X}(y_i \,|\, \boldsymbol{x}_i) \lg \left(p_{Y|X}(y_i \,|\, \boldsymbol{x}_i)\right) - \sum_{m=1}^{M} p_Y(m) \lg \left(p_Y(m)\right) \,. \tag{4.10}$$

For systems simulated using perturbation and proximity techniques, the modified conditional $p_{Y_\epsilon|X}(\cdot)$ in eq. (4.8) and $p'_{Y|X}(m \,|\, \boldsymbol{x})$ in eq. (4.9), along with the unchanged marginal on Y, are used to calculate the ground-truth MI. Thus, this formula provides an estimate of the actual ground-truth MI for the generated system dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, ensuring accurate comparisons between different MI estimation methods.

4.2. Experimental Setup

This section outlines the empirical evaluation process for the proposed MI estimation methods compared to the baselines on various synthetic datasets, as illustrated in Figure 4.3. The primary goal is to assess the approaches' *generalization* capabilities compared to the baselines. The evaluation criteria encompass various factors, including the number of classes (M), input dimensions (d), class imbalance (r), and noise level (ϵ) within synthetic datasets with known MI. The synthetic systems generating datasets are simulated using perturbation and proximity techniques for different unique configurations of $\mathsf{g}_r, d, M, r, \epsilon$, as outlined in Table 4.1.

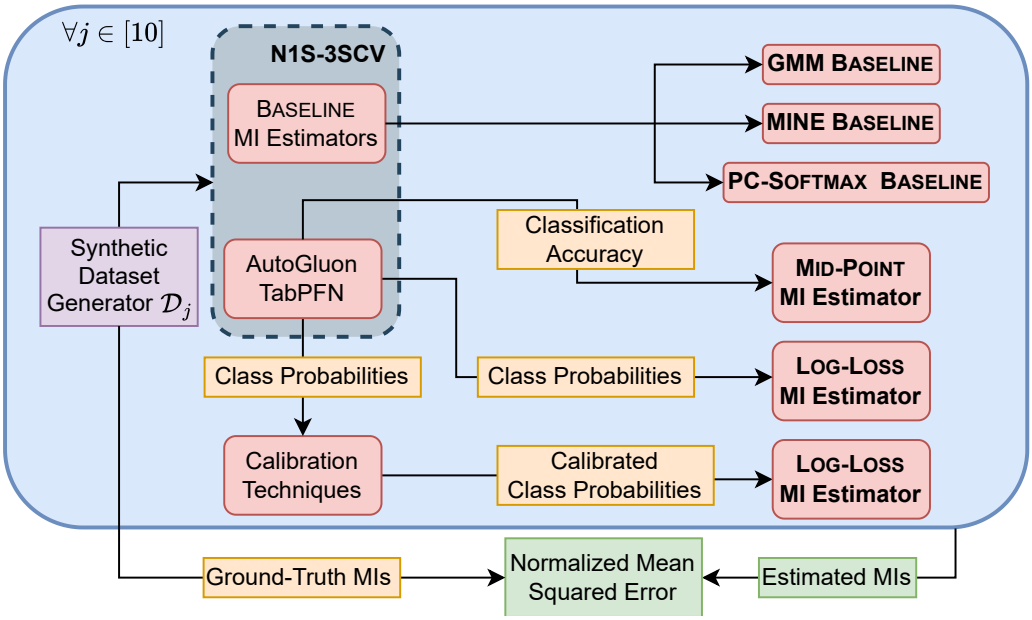


Figure 4.3.: Experimental setup for evaluating MI estimation methods

4.2.1. Evaluation Process

To conduct the experiments, each MI estimation method (detailed in Section 3.3) is applied on synthetic datasets outlined in Table 4.1. For each unique configuration $(\mathbf{g}_r, M, d, r, \epsilon)$, 10 synthetic datasets are generated, denoted as $\mathcal{D}_j$ using different seeds $j \in [10]$. Each approach is applied to these 10 datasets to evaluate their performance using NMAE.

The nested cross-validation with hyperparameter optimization (HPO) is employed for accurate MI estimation on each dataset $\mathcal{D}_j$. The objective functions employed for HPO include balanced error-rate (BER) for PC-SOFTMAX, AutoGluon, and TabPFN, Akaike information criterion (AIC) for Gaussian mixture model (GMM), and mean squared error (MSE) for mutual information neural estimation (MINE) with specific parameter ranges provided in Table A.1. The dataset $\mathcal{D}_j$ is split into a training dataset comprising 70 % of instances and a test dataset containing the remaining 30 %. HPO is then performed with 100 function evaluations using Monte Carlo cross-validation (MCCV) with 3

splits on the training dataset, reserving 30 % of the instances for validation. This process is denoted as "N1S-3SCV" indicating 1 split for test evaluation and 3 MCCV splits for HPO using nested cross-validation (denoted by N), as depicted in Figure 4.3. The AutoGluon is run 30 minutes with BER as an objective using the training dataset. The best-performing model or pipeline is chosen based on the objective function with minimum validation loss (or maximize validation accuracy).

The best model or pipeline is used to estimate MI ($\hat{I}_j$) using the complete dataset $\mathcal{D}_j$ and compare it to the corresponding ground-truth MI (I_j) evaluated using eq. (4.10). The approaches are evaluated using the NMAE metric in eq. (4.11), by comparing the actual MIs ($\boldsymbol{I}$) with the estimated MIs ($\hat{\boldsymbol{I}}$).

Table 4.1.: Overview of the synthetic datasets for MI estimation experiments

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

4.2.2. Evaluation Metric

The generalization performance is assessed using a normalized mean absolute error (MAE), as MI values range from 0 to $\lg(M)$. The MAE is normalized using the entropy of Y: $H_Y(\mathsf{g}_r, M, r) = -\sum_{m=1}^{M} p_Y(m) \lg\left(p_Y(m)\right)$. For each approach and dataset configuration $(\mathsf{g}_r, d, M, r, \epsilon)$, the 10 ground truth MI values are computed using eq. (4.10), denoted as the vector $\boldsymbol{I} = (I_1, \ldots, I_{10})$, and the estimated MI values as $\hat{\boldsymbol{I}} = (\hat{I}_1, \ldots, \hat{I}_{10})$. The performance is evaluated using the normalized MAE (NMAE) metric as:

$$\mathrm{m}_{\mathrm{NMAE}}(\boldsymbol{I}, \hat{\boldsymbol{I}}) = \frac{1}{10} \sum_{j=1}^{10} \frac{|\boldsymbol{I}[j] - \hat{\boldsymbol{I}}[j]|}{H_Y(\mathsf{g}_r, M, r)} = \sum_{j=1}^{10} \frac{|I_j - \hat{I}_j|}{10 \cdot H_Y(\mathsf{g}_r, M, r)} \,. \tag{4.11}$$

4.3. Results

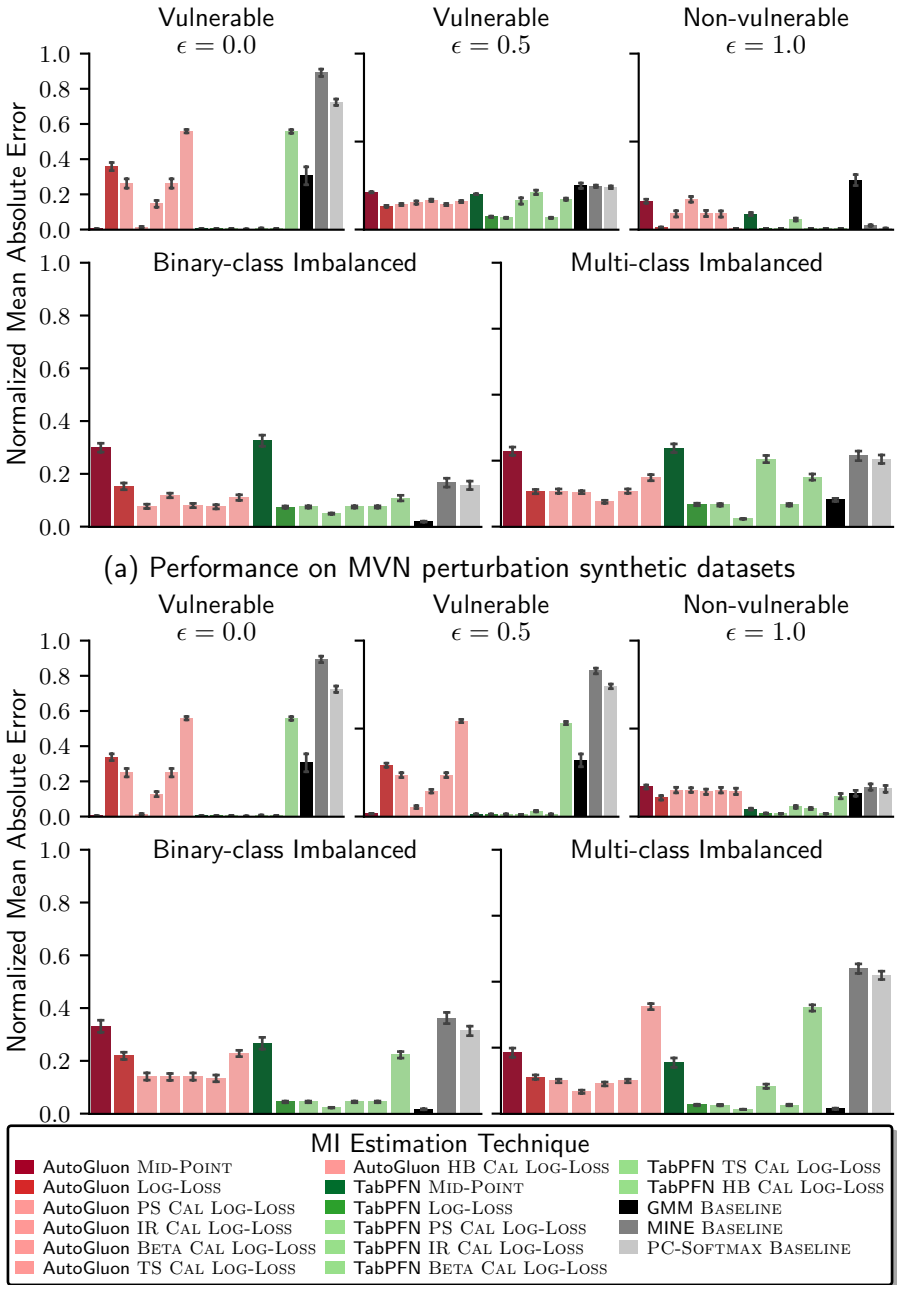
This section provides a comprehensive analysis of the results on MI estimation methods using MVN synthetic datasets, as detailed in Table 4.1. The overall performance of various approaches is discussed on system datasets generated with perturbation and proximity techniques in Section 4.3.1. Additionally, the generalizability of top-performing approaches using AutoGluon and TabPFN including the baselines is assessed across various factors, including the number of classes (M), input dimensions (d), class imbalance (r), and noise levels (ϵ) in Section 4.3.2. For a complete overview of generalization capabilities across all techniques, please refer to Appendix A.4.

4.3.1. Overall Results

The performance of various MI estimation methods is depicted using bar charts displaying the mean and standard error (SE) of NMAE on systems simulated by MVN perturbation and proximity techniques, as shown in Figure 4.4. The analysis includes balanced, binary-imbalanced, and multi-class imbalanced datasets, as outlined in Table 4.1, with scenarios for both no noise and 50 % noise. Additionally, the importance of accurate MI estimation is highlighted for both vulnerable and non-vulnerable systems ($\epsilon = 1.0$) in enhancing ILD performance of the MI-based approaches, discussed in Section 3.2.2.

MVN Perturbation Datasets

The results in Figure 4.4a provide insights into the performance of MI estimation methods applied to synthetic datasets generated using the MVN perturbation technique. In particular, balanced classification datasets highlight the strong performance of the Log-Loss estimation method, especially when TabPFN is combined with appropriate calibration techniques (Cal Log-Loss). TabPFN IR Cal Log-Loss consistently excels in estimating MI (NMAE approximately



(a) Performance on MVN perturbation synthetic datasets

(b) Performance on MVN proximity synthetic datasets

Figure 4.4.: Overall NMAE of MI estimation methods on synthetic dataset

0.003), outperforming AutoGluon, particularly on balanced datasets. For balanced datasets, TabPFN IR Cal Log-Loss achieves a low mean NMAE of approximately 0.0026 for $\epsilon = 0.0$ and 0.0056 for $\epsilon = 1.0$, with low SE values. However, for more vulnerable systems with noise ($\epsilon = 5.0$), IR Cal Log-Loss's performance decreases (NMAE $\approx 0.2115 \pm 0.0124$), while PS Cal Log-Loss and Beta Cal Log-Loss perform better ($\approx 0.067 \pm 0.0032$). Notably, the HB Cal Log-Loss method does not significantly improve Log-Loss estimation for either automated machine learning (AutoML) tool. For systems generating imbalanced binary-class datasets, the GMM baseline performs exceptionally well ($\approx 0.019 \pm 0.0017$), with Furthermore, TabPFN Log-Loss and IR Cal Log-Loss also achieving good results ($\approxeq 0.05$), with very low SEs. In multi-class imbalanced datasets, TabPFN IR Cal Log-Loss leads with a mean NMAE of 0.0293 ± 0.0013, while the GMM also performs strongly ($\approx 0.10 \pm 0.002$), demonstrating adaptability to imbalanced datasets. Overall, TabPFN models with Cal Log-Loss, particularly IR Cal Log-Loss, consistently demonstrate robust performance across diverse MVN perturbation datasets.

MVN Proximity Datasets

The performance of MI estimation methods on synthetic datasets generated using the MVN proximity technique is depicted in Figure 4.4b. Like the perturbation datasets, TabPFN combined with suitable calibration techniques (Cal Log-Loss) consistently excels, particularly for balanced classification datasets, with TabPFN Log-Loss also exhibiting strong performance. Also, AutoGluon models display larger NMAE values, signifying the superiority of TabPFN for this scenario. Notably, TabPFN IR Cal Log-Loss shows outstanding performance for vulnerable systems, with mean NMAE values of 0.0027 for $\epsilon = 0.0$ and 0.0104 for $\epsilon = 0.5$, with low SE. For non-vulnerable systems, TabPFN PS Cal Log-Loss and TS Cal Log-Loss achieve the best results with mean NMAE around 0.0163.

Across binary-class imbalanced datasets, the GMM baseline continues to lead, with the lowest NMAE of 0.017, and TabPFN IR Cal Log-Loss performs similarly well with very low SE. In multi-class imbalanced datasets, TabPFN IR Cal Log-Loss maintains its strong performance with the lowest mean NMAE of 0.01625. At the same time, the GMM also demonstrates solid results, highlighting its versatility in delivering accurate MI estimates across imbalanced synthetic datasets. Overall, the performance of MI estimation on datasets generated using the MVN proximity technique is slightly better than on those generated using the MVN perturbation technique due to the reduced complexity. As discussed in Section 4.1.2, introducing noise using perturbation techniques makes the generated datasets more complex. However, the trends remain consistent: TabPFN with IR Cal Log-Loss consistently outperforms, while AutoGluon models generally exhibit higher NMAE, indicating the superiority of TabPFN in these scenarios.

Summary

In summary, TabPFN, especially with the IR Cal Log-Loss approach, consistently outperforms in MI estimation across systems using MVN perturbation and proximity techniques. While GMM excels in imbalanced datasets, AutoGluon underperforms compared to TabPFN. Calibration techniques (Cal Log-Loss) generally do not impact TabPFN Log-Loss's performance for balanced datasets, suggesting that TabPFN provides well-calibrated class probabilities. However, calibration techniques enhance AutoGluon Log-Loss's performance in vulnerable datasets, indicating that AutoGluon models suffer from poor calibration, whereas TabPFN's class probabilities are naturally well-calibrated, also observed in Appendix A.4. It is also observed that calibration techniques (Cal Log-Loss) can degrade AutoGluon Log-Loss's performance in non-vulnerable systems, overestimating MI and causing the corresponding ILD approach to produce false positives by detecting non-existent ILs, as detailed in Appendix A.3.2.

4.3.2. Generalization Capability Analysis

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, vulnerable with noise levels 0.0 and 0.5), binary-class, and multi-class imbalanced datasets using NMAE, as detailed in Section 4.3.1. The selection criteria rely on the mean NMAE calculated for each dataset type to determine the best-performing approaches for specific scenarios, including non-vulnerable ($\epsilon = 0.0$) and vulnerable synthetic systems with noise levels $\epsilon = 0.0$ and $\epsilon = 0.5$.

Generalization is assessed based on the number of classes (M), input dimensions (d), class imbalance (r), and noise levels (ϵ). For class-based evaluation, the NMAE over all input dimensions ($d \in \{2, 4, \ldots, 20\}$) is aggregated for each possible number of classes ($M \in \{2, 4, \ldots, 10\}$), and similarly for input dimensions by aggregating across all classes. Similarly, for class imbalance generalization, the NMAE across noise levels ($\epsilon \in \{0.0, 0.1, \ldots, 1.0\}$) is aggregated for each imbalance parameter ($r \in \{0.05, 0.1, \ldots, 0.5\}$ in binary-class datasets and $r \in \{0.02, 0.04, \ldots, 0.2\}$ in multi-class datasets), in contrast noise level generalization is assessed by aggregating NMAE across class imbalance values.

The filtered results, depicted in Figures 4.5 and 4.6, present the mean NMAE for the top-performing MI estimation methods across different scenarios on MVN perturbation and proximity datasets, respectively. The Y-axis shows mean NMAE relative to critical factors—number of classes (M), input dimensions (d), class imbalance (r), and noise level (flip percentage (ϵ) in case of MVN perturbation dataset)—offering a detailed view of generalization capabilities for the selected approaches compared to baselines across varied synthetic dataset configurations.

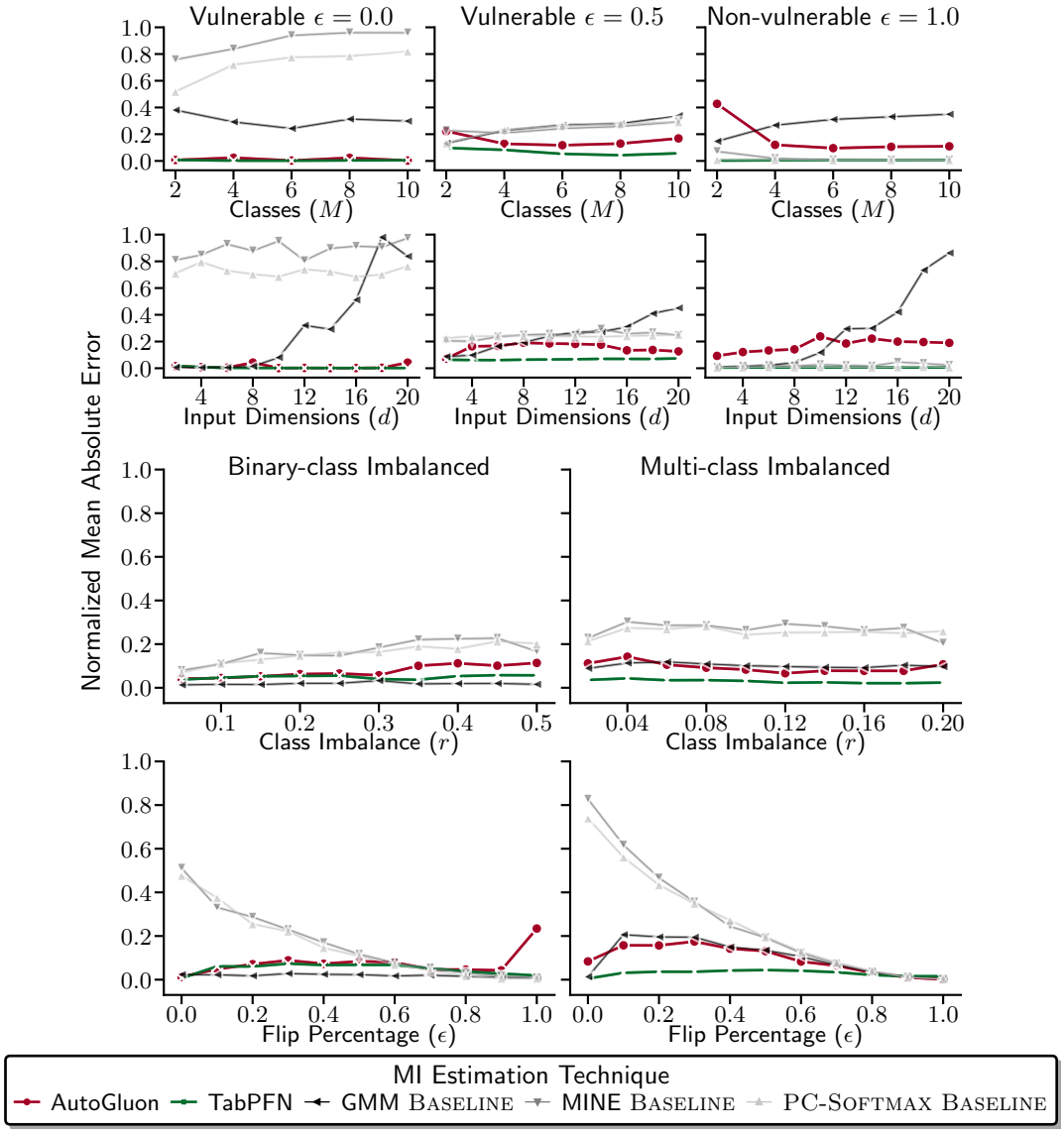


Figure 4.5.: Generalizability of MI estimation methods on MVN perturbation synthetic datasets

Number of Classes (M)

The top row of Figures 4.5 and 4.6, displays the generalization capabilities of MI estimation methods across datasets with increasing the number of classes ($M \in \{2, 4, \ldots, 10\}$) on the X-axis, covering results on datasets generated by systems simulated using MVN perturbation and proximity techniques, respectively.

In datasets generated by ***vulnerable systems***, TabPFN consistently outperforms AutoGluon in MI estimation (showing lower NMAE), with accuracy improving as the number of classes increases. AutoGluon also shows notable gains, particularly for perturbation-based systems at 50 % noise ($\epsilon = 0.5$). Among baselines, MINE and PC-Softmax exhibit the poorest performance as the number of classes increases, with MINE being the least accurate, as illustrated in Figures A.4 and A.5. The GMM baseline occasionally improves (mean NMAE decreases), particularly for multi-class balanced synthetic datasets, though overall, as the class count in vulnerable datasets rises, GMM precision tends to decline due to information loss from aggregation, as confirmed in Figures A.4 and A.5.

For ***non-vulnerable systems***, both TabPFN and AutoGluon show improved MI estimation (lower mean NMAE) as the number of classes grows, with TabPFN consistently outperforming AutoGluon. Baseline performance deteriorates with rising the number of classes, particularly for GMM for datasets generated by non-vulnerable perturbation systems. As an exception to estimating MI in datasets generated by *vulnerable systems*, MINE and PC-Softmax baselines perform better than GMM, particularly for perturbation systems.

Input Dimensions (d)

The second row of Figures 4.5 and 4.6, displays the generalization capabilities of MI estimation methods across datasets with increasing input dimensions ($d \in \{2, 4, \ldots, 20\}$) on the X-axis, covering results on datasets generated by systems simulated using MVN perturbation and proximity techniques, respectively.

In datasets generated by both ***vulnerable*** and ***non-vulnerable*** systems, AutoGluon and TabPFN improve notably with more dimensions, with TabPFN leading and AutoGluon exhibiting more significant improvements, specifically in ones simulated using perturbation technique with 50 % noise. Baseline performance, especially for GMM, declines significantly with increasing input dimensions, especially from 10 to 20 for both systems simulated using perturbation and proximity techniques. For datasets generated by ***vulnerable*** systems, MINE and PC-Softmax baselines also show deterioration in performance most of the time with increasing input dimensions, which becomes significant beyond 10, also illustrated in Figures A.4 and A.5. While MINE and PC-Softmax baselines perform relatively well with increasing input dimensions in datasets generated by ***non-vulnerable*** systems simulated perturbation technique, their estimation accuracy declines for proximity systems. These findings underscore learning challenges due to the curse of dimensionality even when using deep multi-layer perceptrons (MLPs) in MI estimation.

Class Imbalance (r)

The third-row of Figures 4.5 and 4.6 displays the generalization capabilities of MI estimation methods 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 consistently demonstrate high MI estimation accuracy, especially in synthetic datasets created using perturbation techniques, and remain unaffected by increasing imbalance levels.

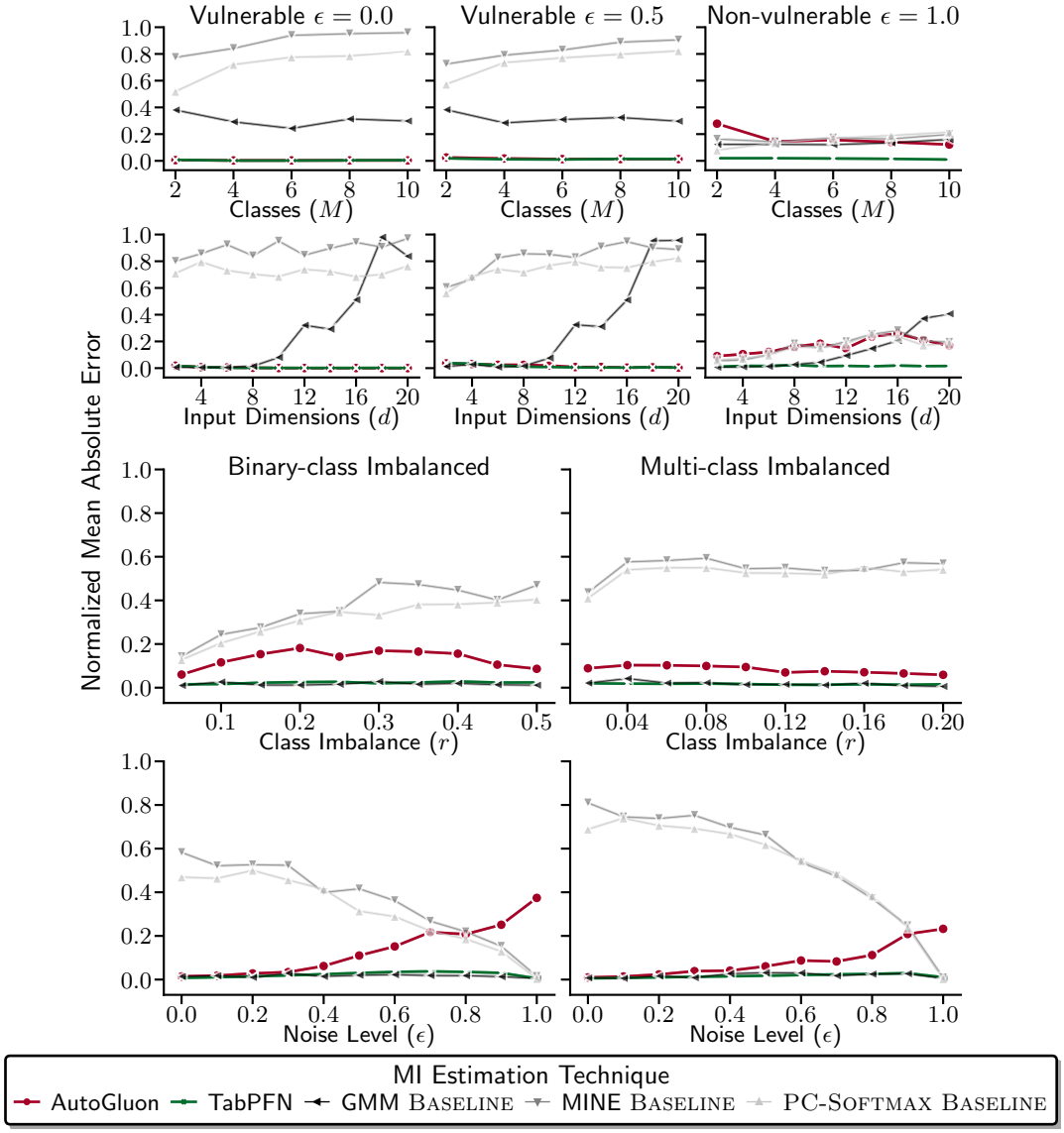


Figure 4.6.: Generalizability of MI estimation methods on MVN proximity synthetic datasets

For systems generating *binary-class imbalanced* datasets, TabPFN and GMM maintain robust performance, with GMM slightly outperforming TabPFN in MVN perturbation datasets. However, AutoGluon, MINE, and PC-Softmax deteriorate, especially beyond a 0.25 imbalance level, performing better in moderately imbalanced datasets yet underperforming compared to TabPFN and GMM overall. In systems generating *multi-class imbalanced* datasets, both TabPFN and GMM continue to perform well, with TabPFN holding a slight edge over GMM in MVN perturbation datasets. Interestingly, AutoGluon improves as class imbalance decreases. However, MINE and PC-Softmax remain stable with a notable decline at a 0.04 imbalance level, highlighting their distinct performance across binary and multi-class imbalanced settings.

Noise Level (ϵ)

The last row of Figures 4.5 and 4.6, generalization capability of MI estimation methods across datasets with varying noise level ($\epsilon \in [0.0, 1.0]$) is depicted on the X-axis, in imbalanced datasets generated using MVN perturbation and MVN proximity techniques, respectively.

In systems generating *binary-class imbalanced* datasets, TabPFN and GMM demonstrate high MI estimation accuracy across all noise levels, with GMM slightly outperforming TabPFN in perturbation-based datasets. In contrast, AutoGluon deteriorates as noise increases, while MINE and PC-Softmax baselines improve substantially, particularly in high-noise scenarios. For systems generating *multi-class imbalanced* datasets, TabPFN and GMM again maintain robust performance across increasing noise levels, with TabPFN slightly outperforming GMM in perturbation datasets. Meanwhile, AutoGluon's performance declines, whereas MINE and PC-Softmax baselines improved with increasing noise in the dataset. These findings underscore the stability and reliability of TabPFN and GMM baseline for estimating MI in systems generating noisy imbalanced datasets.

Summary

This section summarizes the key takeaways regarding the generalization capabilities of the chosen approaches for balanced and imbalanced MVN perturbation and proximity synthetic datasets.

Number of Classes (M) and Input Dimensions (d) The NMAE performance analysis consistently demonstrates strong generalization capabilities for TabPFN and AutoGluon. In contrast, the baselines struggle as both dimensions increase. In particular, GMM shows significant performance degradation beyond 10 input dimensions due to its limitations in high-dimensional spaces, as discussed Section 3.3.3. MINE exhibits the lowest generalization, suggesting its unsuitability for estimating MI between multi-dimensional inputs and one-dimensional outputs in classification datasets. Detailed analysis is provided in Appendix A.4.1.

Class Imbalance (r) and Noise (ϵ) TabPFN and GMM demonstrate strong generalization across varying class imbalances (r) and noise levels (ϵ) in imbalanced datasets. This suggests the GMM 's performance remains effective only in low-dimensional datasets ($d = 5$) and deteriorates with higher dimensions. In contrast, AutoGluon, MINE, and PC-Softmax baselines show weaker generalization in response to class imbalances and noise. Detailed analysis is provided in Appendix A.4.2.

In **conclusion**, TabPFN consistently demonstrates exceptional generalization in MI estimation across multiple factors in both MVN perturbation and proximity datasets.

5. Automating ILD in OpenSSL TLS Servers

This section provides a comprehensive evaluation of information leakage detection (ILD) approaches aimed at detecting side channel leaks (information leakages (ILs)) through time delays and alert or error messages to counter Bleichenbacher attacks on OpenSSL TLS servers, detailed in Section 2.3.2. The empirical setup and an overview of the IL-Datasets used in experiments are detailed in Section 5.2. The complete process of generating these datasets is described in Section 5.1. Our results discussed in Section 5.3 conclude that the proposed information leakage detection (ILD) approaches outperform the state-of-the-art in terms of detection accuracy. This chapter is partially based on the work published in Gupta et al. (2024) [81].

5.1. Side Channels in Network Traces

For Bleichenbacher's attack, it is crucial to distinguish between messages with valid padding and those with incorrect padding based on the server's network response patterns. Funke (2022) [68] generated the timing datasets under the supervision of Drees et al. (2021) [49], who also provided the error code datasets and the initial feature extractor [49, 68]. Drees et al. (2021) [49] implemented a Transport Layer Security (TLS) client builds upon TLS-Attacker[3], a Java-based tool for sending customized TLS protocol messages, which supports various protocol manipulations and has already been used to

detect new Bleichenbacher side channels [108, 21]. This client executes hs TLS handshakes ($hs \in \{500, 50000\}$) using a Rivest–Shamir–Adleman (RSA)-based TLS Cipher Suite, applying distinct padding manipulations [49].

The structure of the PKCS#1v1.5 padded ClientKeyExchange (CKE) message provides multiple ways for a handshake to deviate from the specification, referred to as *manipulations*. TLS-Attacker[3] Böck et al. (2018) [21] supports a range of such manipulations, including the vectors, extended to implement manipulations presented in Meyer et al. (2014) [130], and Klíma-Pokorný-Rosa (KPR) Klíma et al. (2003) [107]. While not exhaustive, this set captures most potential errors in the padding of the PKCS#1v1.5 format and can be expanded.

The client randomly selects and applies a specific manipulation to the ciphertext before performing hs TLS handshakes (with hs being a parameter, e.g., 500 or 50000). Each handshake is logged with the corresponding manipulation to facilitate analysis in subsequent stages. For example, the first byte is replaced (which should be 0x00) with a constant (0x17), generating the manipulation "Incorrect first byte" [49]. To simulate a standard-compliant handshake, the pre-master secret (PMS) is replaced with randomness, ensuring that the handshake still fails during the Finished (FIN) message processing. This approach mimics the Bleichenbacher attack's scenario, where the decrypted message has valid padding. Still, the actual PMS does not match the expected value, preventing the handshake from completing successfully. Each manipulation deviates from the correct PKCS#1v1.5 structure in unique ways [49]:

- `Correctly Formatted PKCS#1 Message` (CFPM): Standard-compliant message, the real PMS replaced with a random string of appropriate length.
- `Wrong First Byte (`0x00 `Set To` 0x17`)` (WFB): Replacing the first byte of the message, which should be 0x00, with a non-zero constant (0x17).

- `Wrong Second Byte (`0x02 `Set To` 0x17`)` (WSB): Replacing the second byte of the message (block type), which should be 0x02, with a non-zero constant (0x17).
- `Invalid TLS Version In PMS` (ITV): Setting the TLS version bytes in the payload to an incorrect constant (0x42, a non-existing version).
- `No` 0x00 `In Message` (N00M): Except for the first byte, all other bytes in the padding string (PS) that are 0x00 (particularly the separator byte) are replaced with 0x01.
- 0x00 `In Some PKCS#1 Padding Byte` (00SPB) Replacing some byte of the PS, which should be non-zero, with 0x00.
- 0x00 `In PKCS#1 Padding (First 8 Bytes After` 0x00 0x02`)` (00FPB): Replacing the ninth byte of the PS, which should be non-zero, with 0x00.
- 0x00 `On The Last Position (`$|PMS| = 0$`)` (00LP): Placing the 0x00 separator at the last byte creates a payload of 0.
- 0x00 `On The Next To Last Position (`$|PMS| = 1$`)` (00NLP): Placing the 0x00 separator at the last-but-one byte, creating a payload of size 1.
- `Correctly Formatted PKCS#1 Message` $|PMS| = 47$ (CFPM47): Valid padding for a 47 bytes PMS, which should be 48 bytes long.
- `Correctly Formatted PKCS#1 PMS Message But 1 Byte Shorter` (CFPM1BS): An otherwise correctly padded message, but with the total length being one byte too short (not matching the RSA modulus size).

To evaluate a broader range of attack scenarios, the two TLS handshake workflows were implemented: a "full" handshake consisting of CKE, ChangeCipherSpec (CCS), and FIN messages, and a "shortened" workflow involving only the CKE to trigger specific vulnerabilities identified in the ROBOT attack [21]. This ensures that different server behaviors are captured under various manipulated ciphertexts. Each handshake's workflow and manipulation are randomly selected to maintain dataset diversity and prevent ordering-based information leakage. However, in the shortened workflow, the missing CCS

and FIN messages can result in server connection timeouts, limiting client throughput. To handle this, Drees et al. (2021) [49] set a one-second timeout for local experiments and a three-second timeout for remote servers to make it suitable for handling varying network delays and ensuring the responses from TLS libraries were captured without missing critical packets.

Feature Extraction The data obtained is recorded using `tcpdump`, resulting in a `.pcap` file containing all messages in their original binary format and metadata. This raw network data is processed using the Python library `pyshark`, which interfaces with Wireshark to transform the messages into real-valued feature vectors and label them with the padding manipulation applied by the client in this handshake to make it compatible with machine learning (ML) classification approaches [49].

Each handshake is represented as an d-dimensional vector, with the manipulation type as the class label. Since the TLS client applies random manipulations, Drees et al. (2021) [49] captures a balanced dataset across all classes. Each handshake's messages are grouped into a single vector, ensuring consistent ordering for accurate pattern recognition. A known challenge is the *curse of dimensionality* [94, 187], where high-dimensional features degrade ML performance. To address this, all messages sent by the server before the manipulated CKE are discarded, and responses relevant to the manipulation are retained and generate a dataset with at least 5 instances for each dimension, which effectively minimizes feature space while preserving the information needed for attack detection. Additionally, only the Transmission Control Protocol (TCP) and TLS layer features are extracted, discarding redundant lower-layer information, making the experiments feasible with limited resources [128]. The decision is based on the previous works on side-channel attacks (SCAs) [128, 21], which only detected behavioral differences on the TCP layer and above.

Since Bleichenbacher's SCAs exploits server behavior to differentiate correctly formatted decrypted messages from incorrect ones [20, 108, 21]. For this thesis, Drees et al. (2021) [49] generated two types of vulnerabilities: one based

on distinct alert messages (error codes) and another on timing variations in response to padding manipulations, as discussed in Sections 5.1.1 and 5.1.2, respectively. Initially focused on alert messages in Gupta et al. (2022) [80], Gupta et al. (2024) [81] later expanded to timing-based side channels, using a dataset with precise measurements to capture subtle server processing time variations from different padding manipulations, as described in Section 5.1.1. However, incorporating timing-based features poses challenges: a non-constant-time client implementation, like TLS-Attacker, could unintentionally leak manipulation details, leading to false positives.

To address this, Drees et al. (2021) [49] refined feature extraction using the implementation in AutoSCA-tool[4] [15] to filter out timing features that could leak client-side behavior. Later, Funke (2022) [68] enhanced AutoSCA with high-precision timing cards, enabling it to detect side channels down to 30 µs on local networks. Our experiments combined error code and timing-based side channels, yielding a comprehensive view of vulnerability detection. However, like Funke (2022) [68], no new side channels were revealed in widely-used open-source TLS libraries.

5.1.1. OpenSSL TLS Timing Datasets

Funke (2022) [68] target side channel vulnerabilities in cryptographic software to validate the approaches using network traffic from an OpenSSL TLS server. A secure server exhibits no time difference when processing correctly and incorrectly formatted messages, ensuring no leakage of secret information. In contrast, a vulnerable OpenSSL TLS server reveals secret information through processing time differences or delays, which Denis leverages for testing the approaches. For the experiments, the timing side channel or time delay, i.e., observable differences in server computation times when processing messages with correct and manipulated padding, is introduced as per the Java TLS implementation

vulnerability `CVE-2014-0411`[1] [40]. Note that the DamnVulnerableOpenSSL[2] server is built upon the OpenSSL 1.0.2l[3], which initially contained multiple intentional side channels, which were removed to in turn add a new patch to include an artificial time delay. Datasets were provided by Funke (2022) [68] from the vulnerable DamnVulnerableOpenSSL[2] TLS server, which exhibits longer computation times for incorrectly formatted messages with manipulated padding and non-vulnerable OpenSSL 1.0.2l[3] TLS server, 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 to incorrectly formatted messages. The IL-Datasets generated for time delays of $t \in \{2^0, 2^1, \ldots, 2^8\} \cup \{5, 10, \ldots, 35\}$ (in micro-seconds, μs) serving as the system's leakage assessment score (LAS), and class imbalances of $r \in \{0.1, 0.3, 0.5\}$ uploaded on OpenML[4], are detailed in Table 5.1. Padding manipulations with delay include: CFPM1BS, 00FPB, N00M, WFB, and WSB, while the ones without delay include: 00SPB, 00LP, 00NLP, CFPM47, and ITV.

5.1.2. OpenSSL TLS Error Code Datasets

side channel vulnerabilities in cryptographic software are analyzed through network traffic from a modified OpenSSL TLS server that leaks secret information via error codes in alert messages (ACK messages). A TLS server returns identical alert responses for correctly and incorrectly formatted messages. However, a vulnerable server responds differently, for example, using an alert message like `handshake failure` instead of `bad record mac` or issuing a TCP disconnect for incorrectly formatted messages. Additionally, vulnerability depends on whether a full handshake occurs, as certain imitation servers are only vulnerable

[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/`

to incomplete handshakes. This behavior is a key feature in the error code datasets, making these types of ILs easier to detect using the ILD approaches, as discussed in Section 5.3.2. Detecting vulnerabilities through alert messages becomes challenging due to subtle timing issues in older OpenSSL versions. Although one such vulnerability was only fixed in 2022, these timing differences are very small ($< 10\mu$s)—rendering them undetectable in the experiments.

The analysis includes OpenSSL-1.1.1k, DamnVulnerableOpenSSL, OpenSSL-0.9.7a, OpenSSL-0.9.7b, and OpenSSL-1.1.1t, each implemented using the TLS-Docker-Library[4]. These servers are example TLS implementations from the OpenSSL source code, compiled from public releases and run within Docker containers to provide various vulnerability profiles. DamnVulnerableOpenSSL, for instance, represents a legacy OpenSSL version with intentionally patched vulnerabilities, helpful in examining known side channel weaknesses in TLS configurations. The latest OpenSSL version 1.1.1k[5] is used as the baseline for a non-vulnerable server implementation, as it is considered secure against Bleichenbacher padding oracle attacks due to extensive scrutiny from the open-source community and security researchers. Similar to OpenSSL-1.1.1k[5], OpenSSL-1.1.1t[6] is widely regarded as resilient against padding oracle or the Bleichenbacher's attacks, serving as a secure baseline due to community scrutiny and continuous updates.

The baseline for a vulnerable server is DamnVulnerableOpenSSL[2], a TLS implementation intentionally crafted with a padding oracle vulnerability for experimental purposes.DamnVulnerableOpenSSL returns distinct TLS alerts for invalid paddings (e.g., `handshake failure` instead of `bad record mac`), confirming its feedback accuracy. Our approach also identifies subtle error-handling differences: upon padding failure, the server disconnects the TCP connection differently, sending a TCP reset immediately after the TCP finished message instead of waiting for client acknowledgment [49, 80]. Consequently,

[4] `https://github.com/tls-attacker/TLS-Docker-Library`
[5] `https://mta.openssl.org/pipermail/openssl-announce/2021-March/000197.html`
[6] `https://mta.openssl.org/pipermail/openssl-announce/2023-February/000249.html`

this sets the `reset` (RST) TCP flag on the second disconnect message only for invalid padding, appearing as a significant feature for classification. The acknowledgment counter shifts forward in the TCP reset case, showing one fewer acknowledgment in contrast to the regular TCP finished message.

Imitating KPR side channels To detect real-world side channels in older implementations, the OpenSSL changelog[7] revealed several vulnerabilities via side channels in prior versions. Initial countermeasures against Bleichenbacher's attack were introduced before version 0.9.5, following its 1998 publication, but were insufficient, as padding failure errors were not adequately masked. These countermeasures were even accidentally removed in version 0.9.5, with version 0.9.6b (2001) later containing the first effective protection against Bleichenbacher's attack. Unfortunately, the source code of these versions is no longer available for download.

Further changes in versions 0.9.6j and 0.9.7b (2003) addressed the recently discovered KPR bad version oracle side channel vulnerability, as detailed in Klíma et al. (2003) [107], the Tenable plugin report[8], and `CVE-2003-0078`[9].

Versions 0.9.7a (vulnerable to KPR with `CVE-2003-0078`[9]) and 0.9.7b (not vulnerable) are available and can be compiled and analyze the approach when faced with a bad version side channel. In the analysis, version 0.9.7a displayed side channel behavior during TLS handshakes with manipulated CKE messages, which was correctly detected by the AutoSCA tool[4] [49]. Testing with version 0.9.7b (non-vulnerable) confirmed that applied countermeasures successfully prevented these side channels, as the AutoSCA tool[4] returned a "not vulnerable" result.

[7] `https://www.openssl.org/news/changelog.html`
[8] `https://www.tenable.com/plugins/nessus/199565`
[9] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0078`

Imitating ROBOT side channels The ROBOT paper by Böck et al. (2018) [21] revealed numerous ILs via real-world side channels across public web servers, especially in closed-source TLS server implementations. Vendors of affected devices were informed, leading to software updates that likely closed these vulnerabilities in most web-facing TLS servers.

To confirm the efficacy of the AutoSCA tool for large-scale internet scans and tested it on Alexa Top 500 domains, with each domain undergoing 500 initial handshakes [49]. Five domains indicated possible vulnerabilities in Drees et al. (2021) [49]; however, further testing with 50 000 handshakes revealed that four were false positives, while cnblogs.com was genuinely vulnerable. This was verified by other scanners, including ssllabs[10], tls-scanner[11], and testssl.sh[12], which confirmed the side channel presence. The server operators were contacted and notified of the issue.

Our approach scales to hundreds of web servers, albeit with a higher runtime than other ROBOT scanners. Unlike these, the AutoSCA tool[4] assumes less about the specific nature of the side channel, thus proving its real-world applicability where network behavior variably affects traffic traces.

Most ROBOT side channels have been removed from prominent servers, so they are re-implemented using Mbed-TLS [13] configurations, mimicking four ROBOT vulnerabilities that cover a representative set of server behaviors as

Facebook F5 v1 (`CVE-2017-6168`[14]) and Facebook v2 use a single CKE message, causing a TCP timeout if the padding is invalid, as the server waits for CCS and FIN messages. For invalid paddings, these servers do not wait but abort prematurely with a TLS alert or a TCP disconnect (TCP finished message).

[10] `https://www.ssllabs.com/ssltest`
[11] `https://github.com/tls-attacker/TLS-Scanner`
[12] `https://testssl.sh/`
[13] `https://github.com/Mbed-TLS/mbedtls`
[14] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6168`

Cisco Cisco ACE (`CVE-2017-17428`[15]) returns a TLS alert 47 instead of 20 when padding fails.

NetScalerGCM Citrix NetScaler (`CVE-2017-17382`[16]) provides a TLS alert 51 for valid padding but times out for invalid padding, exposing a padding oracle vulnerability.

PAN-OS PAN-OS (`CVE-2017-17841`[17]) sends a TLS alert 40 in both cases but duplicates the alert when padding fails.

5.2. Experimental Setup

This section outlines the empirical process for all ILD approaches, including the baselines for detecting timing side channel leaks in OpenSSL TLS servers, as depicted in Figure 5.1. Our main objective is to assess the generalization capability of various ILD approaches discussed in Section 3.2.2 across different class imbalances ($r \in \{0.1, 0.3, 0.5\}$) in the system dataset using detection accuracy. Additionally, the generalization capability of these approaches is assessed with respect to the complexity of IL in the system, quantified by the time delay of the OpenSSL TLS server, refer to Appendix A.3 for details.

Table 5.1.: Overview of the OpenSSL TLS timing IL-Datasets used for the ILD experiments.

Time Delay (in micro-seconds,μs)	# Folds	IL-Dataset $\mathcal{L}$ configuration			Dataset $\mathcal{D}$ configuration				
		# Systems $\lvert\mathcal{L}\rvert$	# $z=0$	# $z=1$	Imbalance r	$\lvert\mathcal{D}\rvert$	# $y=0$	# $y=1$	# Features
$\{2^0, 2^1, \ldots, 2^8\}$	3	10	5	5	0.1 0.3 0.5	$[1886, 2128]$ $[1212, 1368]$ $[1725, 1929]$	$[1698, 1916]$ $[849, 958]$ $[849, 958]$	$[188, 212]$ $[363, 410]$ $[829, 989]$	124
$\{10, 15, \ldots, 35\}$	10	10	5	5	0.1 0.3 0.5	$[1924, 3992]$ $[1237, 2910]$ $[1721, 4084]$	$[1732, 3594]$ $[866, 2037]$ $[866, 2037]$	$[192, 398]$ $[371, 873]$ $[826, 2104]$	$[124, 154]$

[15] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17428`
[16] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17382`
[17] `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17841`

To achieve this, multiple IL-Datasets generated by Funke (2022) [68] with time delay values ranging from 1 to 256 and class imbalances $r \in \{0.1, 0.3, 0.5\}$ are used, as described in Table 5.1. Additionally, Funke (2022) [68] generated 9 datasets from different OpenSSL TLS servers, which are vulnerable due to distinct TLS alert messages (e.g., `handshake failure` instead of `bad record mac`) in response to certain manipulated paddings within the network traces, as detailed in Table 5.2.

5.2.1. Evaluation Process

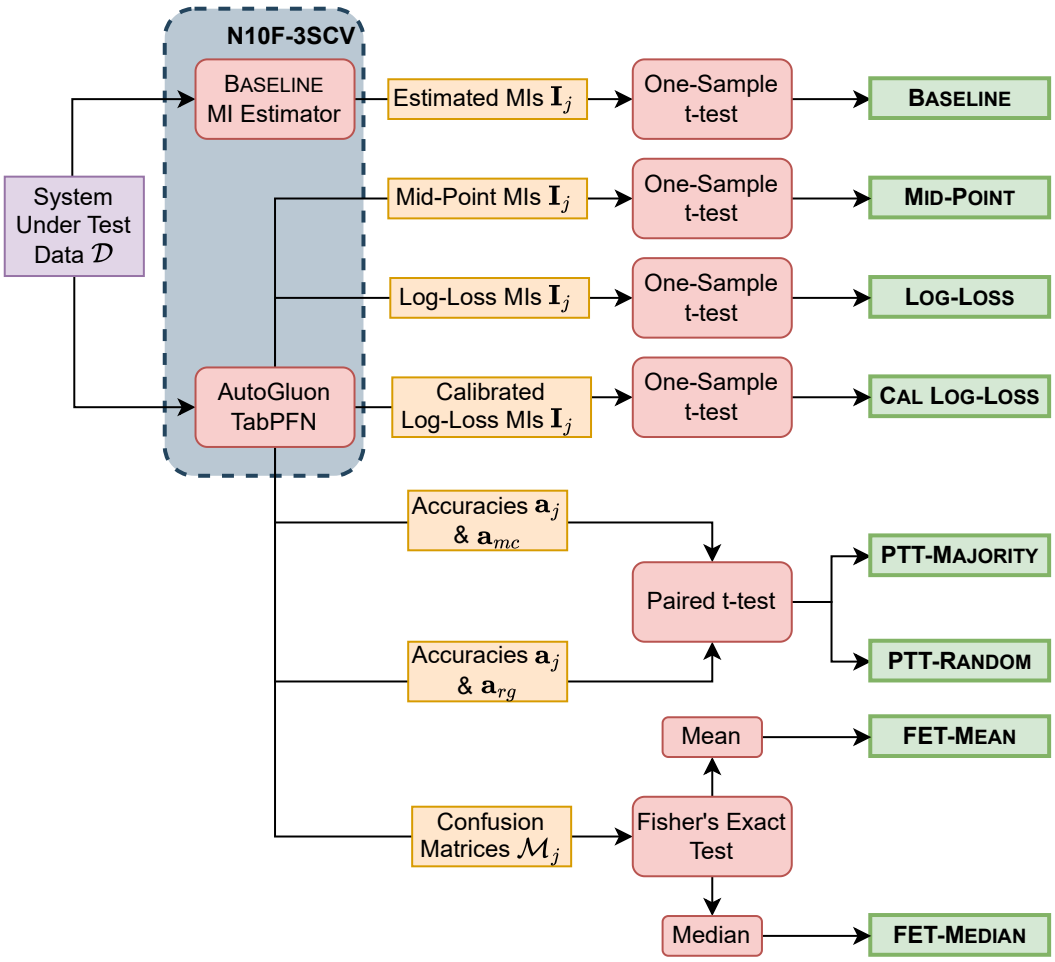
To conduct the experiments, each ILD approach depicted in Figure 5.1a on a set of IL-Datasets detailed in Tables 5.1 and 5.2 are employed. The experiments aim to evaluate the generalizability of ILD approaches under class imbalance and detect ILs of varying magnitudes in the systems via the generated dataset. Each ILD approach employs the detection process with a rejection threshold of $\lfloor \frac{J}{2} \rfloor = 5$ on the cut-off parameter τ.

Methodology The process of detecting IL in a system generating classification dataset by various ILD approaches is depicted in Figure 5.1. To enhance IL detection confidence, the Holm-Bonferroni correction is applied on p-values by using statistical tests (details in Section 2.4) on performance estimates of the top-10 automated machine learning (AutoML) models or pipelines from AutoGluon and TabPFN obtained through hyperparameter optimization (HPO). The ILD process involves rigorous estimation, statistical assessment, and correction techniques to detect IL in a system confidently. The process starts with obtaining accurate mutual information (MI) or Bayes predictor performance estimates using nested K-fold cross-validation (KFCV) with HPO and specific parameter ranges provided in Table A.1. The dataset $\mathcal{D}$ is split into 90 % training and 10 % test sets using K-fold cross-validation (KFCV) (10), conducting HPO with 100 evaluations using Monte Carlo cross-validation (MCCV) with 3 splits, reserving 30 % of training data for validation, denoted

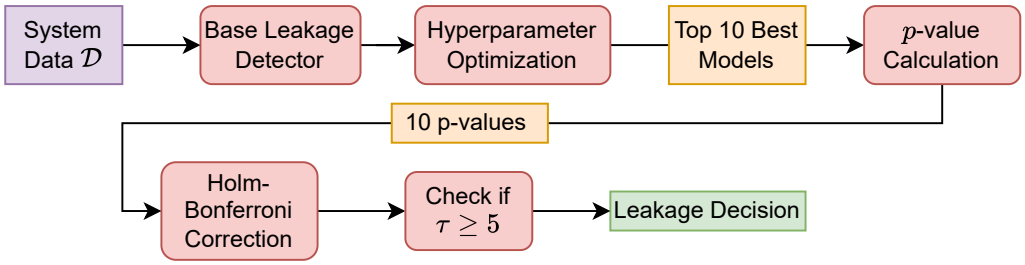
as "N10F-3SCV", as depicted in Figure 5.1a. Objective functions for HPO include balanced error-rate (BER) for PC-Softmax, AutoGluon, and TabPFN, Akaike information criterion (AIC) for Gaussian mixture model (GMM), and mean squared error (MSE) for mutual information neural estimation (MINE) with parameter ranges provided in Table A.1. The top-10 best-performing pipelines are identified 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. The 10 MI estimates, accuracies, and confusion matrices (CMs) are obtained 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 one-sample t-test (OTT) is used on MI estimates($\hat{\boldsymbol{I}}_j$), paired t-test (PTT) compares the accuracies ($\boldsymbol{a}_j$) with that of g_p^{mc} ($\boldsymbol{a}_{mc}$), and Fisher's exact test (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.

The Holm-Bonferroni correction enhances **robustness and reliability** in ILD by mitigating overfitting and noise from single-model estimates [91]. After obtaining 10 p-values from top-10 models/pipelines, the correction is applied 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, as a higher threshold avoids false positives and a lower value reduces false negatives, as detailed in Section 2.4.3. It is concluded in Gupta et al. (2022) [80] to set the rejection threshold to $\lfloor J/2 \rfloor$ on the cut-off parameter τ to ensure robust and accurate IL detection, i.e., $\tau \geq 5$ [80, 81].

Evaluation Each IL-Dataset ($\mathcal{L} = \{(\mathcal{D}_i, z_i)\}_{i=1}^{10}$) consists of 10 different system datasets, each corresponding to a distinct incorrect padding manipulation, as discussed in Section 5.1. Specifically, within an IL-Dataset generated by an OpenSSL TLS server vulnerable to processing times, five systems contain IL



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



(b) Information leakage (IL) detection process

Figure 5.1.: Procedure of using ILD approaches to detect ILs in a systems generating classification dataset $\mathcal{D}$

(i.e., $z = 1$), and five systems do not contain IL (i.e., $z = 0$), as detailed in Section 5.1.1. Additionally, within an IL-Dataset generated by 9 real-world OpenSSL TLS servers vulnerable to error codes with three of them are entirely non-vulnerable servers with all 10 systems do not contain IL (i.e., $z = 0$) and remaining servers being vulnerable to some of the padding manipulations, as detailed in Section 5.1.2. Each ILD approach all IL-Datasets to generate predicted IL decisions denoted by a vector $\hat{\boldsymbol{z}} = (\hat{z}_1, \ldots, \hat{z}_{10})$. These predictions are then compared to the corresponding ground-truth vector $\boldsymbol{z} = (z_1, \ldots, z_{10})$ using standard binary classification metrics, as defined in Section 3.1.

Table 5.2.: Overview of the OpenSSL TLS error code IL-Datasets.

Server	IL-Dataset $\mathcal{L}$ configuration				Dataset $\mathcal{D}$ configuration				
	# Systems $\lvert\mathcal{L}\rvert$	# $z=0$	# $z=1$	Vulnerable Classes	Imbalance r	$\lvert\mathcal{D}\rvert$	# $y=0$	# $y=1$	# Features
OpenSSL TLS Servers with KPR side channels									
DamnVulnerableOpenSSL	10	4	6	00FPB, 00LP CFPM1BS, N00M WFB, WSB	0.1 0.3 0.5	22331 [22331, 28711] [22331, 40221]	20098 20098 20098	2233 [2233, 8613] [2233, 20123]	94
OpenSSL-0.9.7a	10	9	1	ITV	0.1 0.3 0.5	22106 [22106, 28422] [22106, 40272]	19896 19896 19896	2210 [2210, 8526] [2210, 20376]	124
Non-vulnerable OpenSSL TLS servers									
OpenSSL-0.9.7b	10	10	0	Non-vulnerable	0.1 0.3 0.5	22174 [22174, 28510] [22174, 40214]	19957 19957 19957	2217 [2217, 8553] [2217, 20257]	124
OpenSSL-1.1.1t	10	10	0	Non-vulnerable	0.1 0.3 0.5	22100 [22100, 28414] [22100, 40106]	19890 19890 19890	2210 [2210, 8524] [2210, 20216]	94
OpenSSL-1.1.1k	10	10	0	Non-vulnerable	0.1 0.3 0.5	5153 [5153, 6625] [5153, 9291]	4638 4638 4638	515 [515, 1987] [515, 4653]	116
OpenSSL TLS servers with ROBOT side channels									
Cisco	10	6	4	00FPB, 00LP WFB, WSB	0.1 0.3 0.5	1568 [1008, 1568] [1008, 1568]	1412 [706, 1412] [706, 1412]	156 [156, 302] [156, 737]	252
Facebook	10	4	6	00FPB, 00SPB CFPM1BS, N00M WFB, WSB	0.1 0.3 0.5	3884 [2497, 3884] [2497, 3884]	3496 [1748, 3496] [1748, 3496]	388 [388, 749] [388, 1857]	180
NetScalerGCM	10	4	6	00FPB, 00SPB CFPM1BS, N00M WFB, WSB	0.1 0.3 0.5	1616 [1040, 1616] [1040, 1616]	1456 [728, 1456] [728, 1456]	160 [160, 312] [160, 741]	252
PAN-OS	10	4	6	00FPB, 00SPB CFPM1BS, N00M WFB, WSB	0.1 0.3 0.5	1450 [932, 1450] [932, 1450]	1306 [653, 1306] [653, 1306]	144 [144, 279] [144, 747]	196

5.3. Results

This comprehensively analyzes the performance of various ILD approaches in detecting ILs through time delays and error codes in the alert messages to counter Bleichenbacher attacks on OpenSSL TLS servers. The generalization performance of various ILD approaches, the results are analyzed with varying levels of class imbalance in the underlying dataset generated by the system. Additionally, various systems simulated using the OpenSSL TLS servers configured with different values of time delays ranging from 2 to 256 μs (micro-seconds), which can be exploited to perform the Bleichenbacher SCAs. This experiment provides insights into the robustness of ILD approaches in detecting small ILs in the systems.

5.3.1. Detection Accuracy on Timing Datasets

The performance of various ILD approaches is comprehensively analyzed with respect class imbalances ($r \in 0.1, 0.3, 0.5$) and time delays ranging from 2 to 256 μs (in micro-seconds). This study focuses on OpenSSL TLS systems with short time delays (≤ 25 μs), which are associated with complex ILs that are challenging to detect and with significant time delays (≥ 25 μs), where detection is relatively simple. These factors are crucial in understanding how different approaches perform based on class imbalance in the OpenSSL TLS system datasets and the complexity of the IL patterns.

To evaluate the performance of ILD approaches, the mean detection accuracy with standard error (SE) for each method across different IL-Datasets is presented, employing a rejection threshold of 5 on the cut-off parameter τ, as discussed in Section 5.2.1. For a more detailed analysis of generalization capability in terms of Accuracy, FPR, and FNR concerning IL complexity, refer to Appendix A.3.

Selected ILD Approaches To identify the best-performing calibration approach for both TabPFN and AutoGluon using Log-Loss estimation, their performance with respect to normalized mean absolute error (NMAE) is assessed obtained from all experiments detailed in Section 4.3.1. This process evaluated the performance of binary-class synthetic datasets to determine the optimal calibration approach for each AutoML tool, separately for balanced ($r = 0.5$) and imbalanced ($r \in \{0.1, 0.3\}$) datasets.

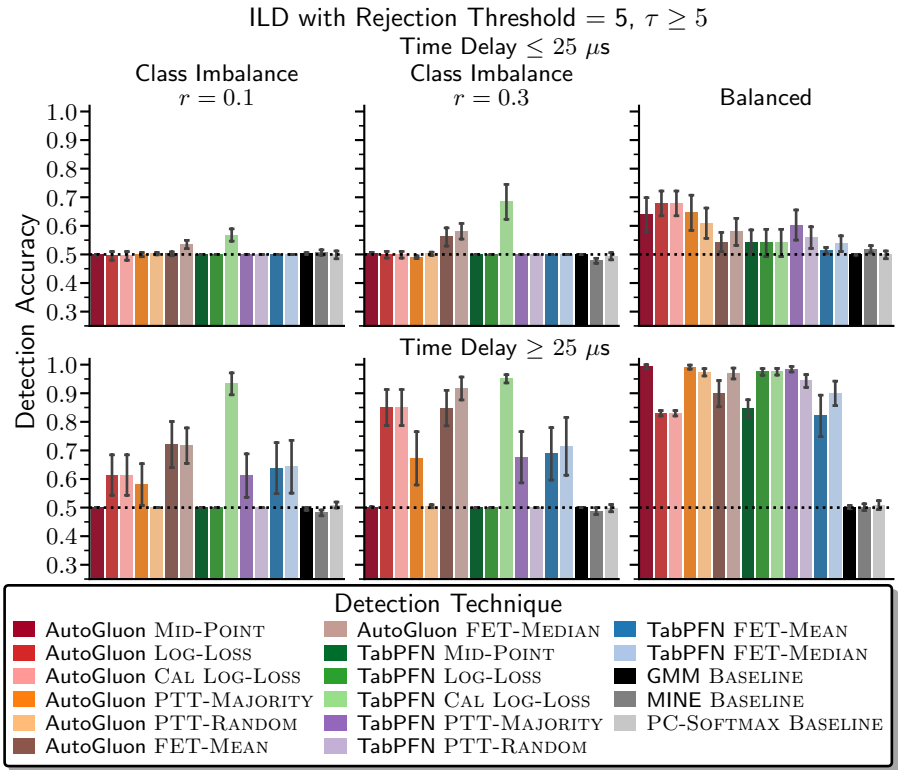


Figure 5.2.: Detection accuracy of ILD approaches in detecting timing side channels in OpenSSL TLS servers

Based on this evaluation, AutoGluon TS Cal Log-Loss is the most effective for both cases, while TabPFN IR Cal Log-Loss performed best on imbalanced datasets, and TabPFN PS Cal Log-Loss excelled in balanced ones.

Short Time Delay

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

Imbalanced Detecting ILs in systems that generate imbalanced datasets with short time delays or LAS proves to be particularly challenging, as most ILD approaches struggle to detect ILs effectively. The TabPFN Cal Log-Loss approach consistently demonstrates higher detection accuracy, while AutoGluon FET-Median also performs reasonably well. For $r = 0.1$, TabPFN Cal Log-Loss detects 69 % of ILs, and AutoGluon FET-Median detects around 54 %. At $r = 0.3$, both TabPFN Cal Log-Loss and AutoGluon FET-Median detect 58 % of ILs. Overall, these scenarios remain challenging due to missed ILs and high false positive rates (FPRs), as detailed in Appendix A.3.

Balanced Detecting ILs in systems generating balanced datasets with short time delays remains challenging but 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 accuracy, as confirmed in Appendix A.3.1. Compared to imbalanced ones, AutoGluon Mid-Point performs better on balanced datasets, achieving around 64 % accuracy. Overall, AutoGluon outperforms TabPFN, with PTT-Majority detecting over 61 % of ILs, outperforming the ones using FET to detect ILs.

Long Time Delay

Our second set of experiments focuses on systems with long time delay (≥ 25 μs) with more straightforward to detect ILs.

Imbalanced Detecting ILs in systems with more significant time delay (≥ 25 μs) or LAS, generating imbalanced datasets becomes more manageable, with all approaches consistently performing well, detecting significantly more than 50 % of ILs. The TabPFN Cal Log-Loss approach, one of the top performers, detected about 94 % of ILs, while AutoGluon FET-Median detecting over 92 % for $r = 0.3$ and around 72 % for $r = 0.1$. It is worth noting that FET based approaches outperform PTT-Majority for imbalanced datasets generated by systems with long time delays, which aligns with the findings for detecting ILs via alert messages in Section 5.3.2.

Balanced In systems with balanced datasets and longer time delays (≥ 25 μs), detecting ILs becomes significantly more straightforward, with most methods achieving high detection rates. AutoGluon Mid-Point and TabPFN IR Cal Log-Loss consistently detect the majority (around 99 %) of ILs, confirming IR Cal Log-Loss's offering an alternative calibration technique to improve the efficacy for TabPFN Log-Loss approach, as confirmed in Appendix A.3.1. TabPFN PS Cal Log-Loss and Log-Loss approaches detect 97 %, showing no improvement with the selected calibration technique, while TabPFN Mid-Point detects 85 %. Amongst the classification-based approaches, PTT-Majority using both AutoML tools consistently outperforms, detecting 99 % of ILs. In contrast, AutoGluon FET-Median detects 96 % and TabPFN FET-Median only 90 % of ILs. This finding differs from the results in Section 5.3.2, where FET-based approaches perform significantly better in detecting ILs via alert messages, while PTT performs better ILD for timing-based SCAs.

5.3.2. Detection Accuracy on Error code Datasets

The performance of various ILD approaches is analyzed using the detection accuracy, FPR, and false negative rate (FNR) in datasets extracted using network traces simulated from 9 OpenSSL TLS servers, which leaks information through TLS alerts or error codes (in ACK messages), making them vulnerable

to Bleichenbacher attacks. The overall detection accuracy results for datasets systems with different class imbalances ($r \in 0.1, 0.3, 0.5$), aggregated across all servers, are summarized in Figure 5.3, with detailed analysis of all ILD approaches on each OpenSSL TLS server in Appendix A.2.

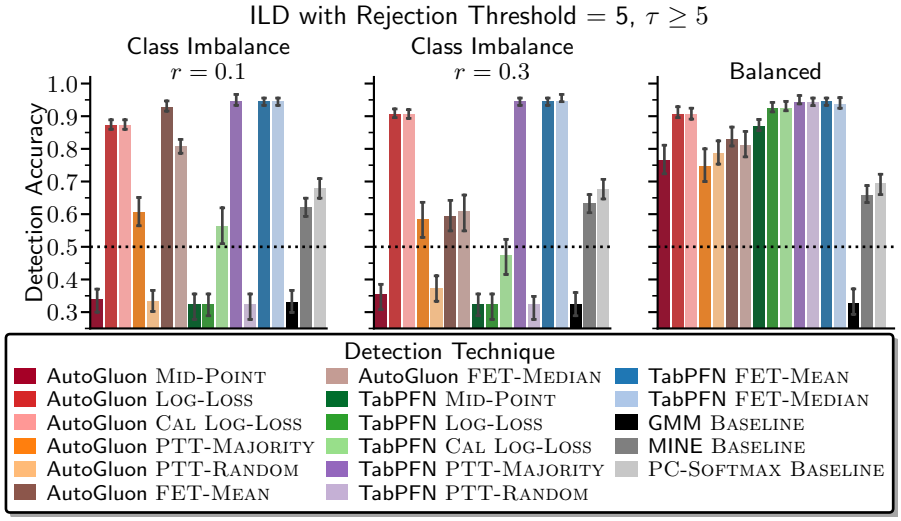


Figure 5.3.: Detection accuracy of Selected ILD approaches in detecting side channels in OpenSSL TLS servers

Imbalanced

When detecting ILs in systems that generate imbalanced datasets, detection accuracy is notably lower compared to ones generating balanced datasets. Among MI-based approaches, Log-Loss and Cal Log-Loss using AutoGluon detected more than 90 % of ILs, outperforming ones using TabPFN, in contrast to findings in Section 5.3.1, where TabPFN Cal Log-Loss is the top performer. Generally, the classification-based approaches FET-Mean, FET-Median, and PTT-Majority worked very well for imbalanced system datasets. As expected, Mid-Point and PTT-Random approaches failed, producing false positives and confirming that Mid-Point suffers from MI overestimation, indicating a fundamental limitation in handling class distribution shifts, as discussed in eq. (3.2) in Section 3.3.1.

The top performers are TabPFN FET-Mean, TabPFN FET-Median, and TabPFN PTT-Majority, each achieving around 95 % detection accuracy. Among classification-based approaches using AutoGluon, FET-Mean achieved 93 % detection accuracy for datasets with a class imbalance of 0.1, but only 60 % for those with an imbalance of 0.3. Interestingly, MINE and probabilitiy-corrected softmax (PC-softmax) detected more than 62 % of ILs, while GMM detected less than 50 %, performing worse than random guessing, a result contrary to the performance observed in Section 5.3.1.

Balanced

Similar to observations in Section 5.3.1, ILD approaches detected more than 90 % of ILs in systems, including all MI-based and classification-based methods using TabPFN, as well as AutoGluon Log-Loss and AutoGluon Cal Log-Loss. Among classification-based approaches using AutoGluon, the ones using FET detected over 80 % of ILs in systems, while those using PTT detect more than 75 % of ILs. Interestingly, MINE and PC-softmax also detected over 62 %, whereas GMM again detected less than 50 %, performing worse than random guessing and contrasting with results in Section 5.3.1. The reason for this is that it overfits due to the overestimation of MI and production of false positives in detecting ILs, as detailed in Appendix A.2, A.3 and A.4.

5.3.3. Summary

Detecting IL is generally easier in systems generating balanced datasets, where FET-Mean, PTT-Majority using AutoGluon, and TabPFN demonstrate strong performance.Unlike observations by Gupta et al. (2022) [80], PTT-based approaches sometimes outperform FET-based ones on balanced datasets, indicating that ILD effectiveness varies with IL patterns.Classification-based approaches FET-Mean, FET-Median, and PTT-Majority perform well

on imbalanced datasets, while Mid-Point and PTT-Random struggle, often producing false positives due to Mid-Point's MI overestimation, a limitation in handling imbalanced datasets noted in Section 3.3.1.

TabPFN Cal Log-Loss consistently outperforms other methods, emphasizing the importance of calibrating TabPFN Log-Loss for precise MI estimation. For OpenSSL TLS timing balanced datasets, IR Cal Log-Loss enhances detection accuracy over PS Cal Log-Loss, as shown in Appendix A.3, aligning with synthetic dataset results in Section 4.3.1. In error code TLS datasets, HB Cal Log-Loss surpasses both PS Cal Log-Loss and IR Cal Log-Loss, as detailed in Appendix A.2, highlighting the need for dataset-specific calibration selection. Feature reduction in TabPFN (from 150 to $[20, 50]$) and scarce positive samples in imbalanced datasets further emphasize the dependence of calibration choice on dataset characteristics. For AutoGluon, Log-Loss calibration often overfits and overestimates MI, causing false positives in OpenSSL TLS timing balanced datasets, as observed in Section 4.3.1. However, in error code datasets, Log-Loss calibration demonstrates robust performance, underlining the critical role of dataset-specific calibration in Log-Loss.

In timing datasets, baseline methods detect only about 50 % of ILs, reflecting random detection patterns, as confirmed in Appendix A.3. Conversely, in error code datasets, GMM detects less than 50 %, performing below random guessing, while MINE and PC-softmax exceed 62 %, contrasting timing dataset results (c.f. Section 5.3.1). MINE and PC-softmax fail to detect ILs, producing false negatives due to MI underestimation, while GMM overfits, overestimating MI and producing false positives, as detailed in Appendix A.3 and A.2. In timing datasets, MI-based approaches using TabPFN outperform those using AutoGluon, while in error code datasets, AutoGluon outperforms TabPFN. This suggests that AutoML tools are suited to specific vulnerabilities: AutoGluon for error code and TabPFN for timing datasets, implying that tailoring and selection of the AutoML tool depends on the underlying system dataset, also confirmed in Appendix A.3 and A.2.

6. Automating ILD in AES-encrypted Systems

This chapter provides an overview of the automated framework using the guessing entropy (GE) and trace sufficiency threshold (TST) values designed to detect and assess information leakage (IL) in AES-encrypted systems, in Section 6.1. Additionally, the experimental setup to analyze the optimal parameters is described, i.e., the search strategy employed for neural architecture search (NAS) and the shape of the input features in Section 6.2, with methodology in Section 6.1.1 and Section 6.2.2 presents the datasets used for the experiments. Implementation details are summarized in Appendix A.1.3. The results of selecting the optimal parameters, IL assessment in terms of vulnerability score (VS) and efficiency in terms of TST, is detailed in Section 6.3. This chapter is based on the work published in Gupta et al. (2023) [79].

6.1. ILD in AES-encrypted Systems

Information leakage detection (ILD) in AES encrypted systems is crucial for understanding how much information about the secret key can be inferred from side channel traces during the encryption process. As Advanced Encryption Standard (AES) is widely employed for securing sensitive data, it becomes an attractive target for side-channel attacks (SCAs), where attackers exploit physical leakages, such as power consumption or electromagnetic radiations

(EMR), to extract secret key information. Given the complexity and multi-round structure of AES encryption, leakage can occur at various stages, making it an ideal subject for ILD techniques, as discussed in Section 2.3.3.

While mutual information (MI) estimation is commonly used to quantify and detect IL, challenges arise with the high-dimensional nature of AES traces and the difficulty in capturing non-linear relationships between leakage and secret key bytes. These include discretization, slow convergence, and bias, particularly in noisy datasets [35]. Recently, deep learning (DL) models have been increasingly applied to perform SCA and detect ILs in AES encrypted systems [124]. One such approach is mutual information neural estimation (MINE), which MI using DL, which allows scalable estimation in high-dimensional spaces, but it requires up to $200,000$ epochs, making the analysis computationally expensive [39, 40]. Moreover, MINE faces slow convergence and is sensitive to architecture tuning, particularly in imbalanced datasets [40]. Similarly, Deep Learning Leakage Assessment (DL-LA) uses DL models to bypass traditional pre-processing steps, such as trace alignment and multivariate leakage modeling, proving effective for detecting ILs in non-profiled SCAs. However, both MINE and DL-LA rely on multi-layer perceptrons (MLPs) for MI estimation, which limits their applicability in AES encrypted systems, as MLPs are often outperformed by convolutional neural networks (CNNs) in detecting side channel leakage [206, 200, 143, 25]. In this field, the most relevant literature tends to use machine learning (ML) or DL for performing SCAs, rather than focusing on early detection of ILs to prevent them [86]. Notably, current DL-based methods detect side channels and protect systems from SCAs at both the algorithmic and hardware levels [137, 136, 39]. Though DL models have shown effectiveness in detecting ILs by analyzing model accuracy on system data, they struggle with imbalanced, noisy real-world datasets, which can lead to missed ILs, producing false negatives [136, 207, 148].

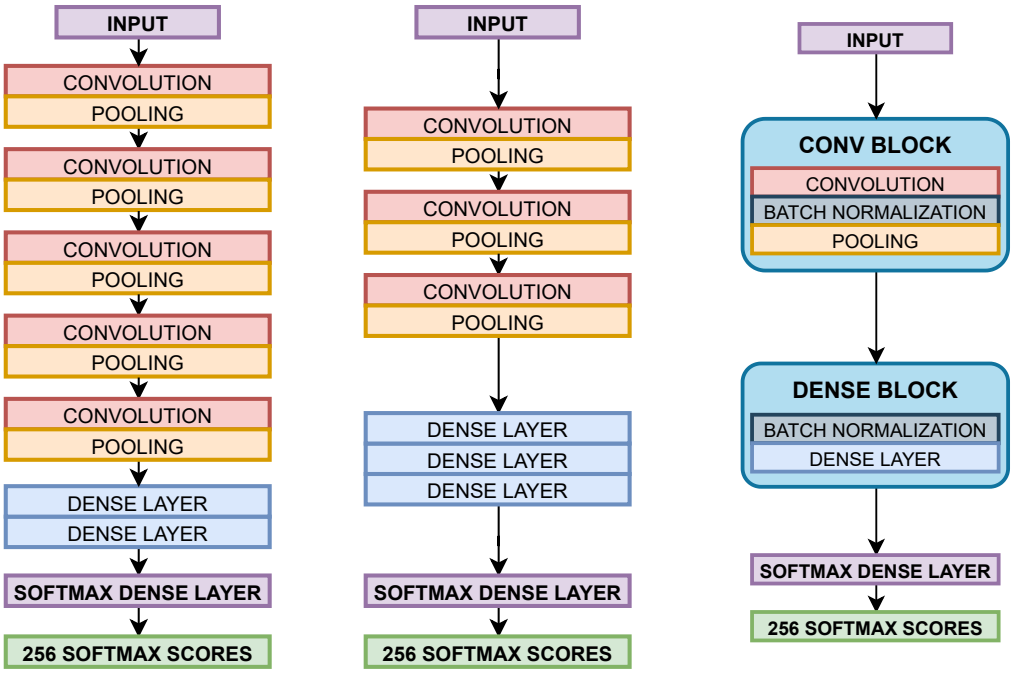
To overcome these limitations, it is proposed to use GE of the learned DL model on the traces acquired from the system as a metric for quantifying IL. GE is directly related to MI and provides a more interpretable measure by

quantifying how many guesses an attacker needs to recover the encryption key, as detailed in Section 3.2.1. Additionally, to address the challenges of noisy traces and architecture tuning, NAS is employed on CNNs, which has been shown to perform better than MLPs in detecting side channel leakage. By automating the ILD process with NAS, multiple optimal models can be generated to analyze various GE values to assess leakage by evaluating the VS, as detailed in Section 3.2.2. This approach also incorporates the analysis of TST values to offer a clear measure of system vulnerability and assesses IL in AES encrypted systems.

6.1.1. Automated side channel Attacks

The first handcrafted CNN architecture is shown in Figure 6.1a, was initially developed by Benadjila et al. (2020) [14] to target the ASCAD dataset, manually refined to create smaller, dataset-specific versions, such as those shown in Figure 6.1b developed by Zaid et al. (2019) [206], which were tailored for attacking ASCAD_f 50ms and ASCAD_f 100ms datasets. This demonstrates that the optimal CNN architecture is highly dependent on the dataset, and manually designing such architectures requires significant expertise. This challenge is not exclusive to SCAs, which has led to the adoption of automated solutions like *Neural architecture search (NAS)*. NAS frames the task of finding an appropriate architecture for a given dataset as an optimization problem based on a defined objective function, allowing it to be solved automatically. Typically, NAS employs an evaluation metric, such as accuracy or a loss function, as its *objective*, which serves as the performance criterion for evaluating architectures, as detailed in Section 2.2.4. The dataset is split into a training set $\mathcal{D}_{train}$, used to train an architecture $A \in \mathcal{A}$, and a validation set $\mathcal{D}_{val}$, used to assess its performance using a chosen *objective*, such as accuracy, which is ultimately used to identify the best-performing architecture as shown in Figure 6.2. This approach inspired the use of NAS in SCAs, where the profiling dataset is provided as input, allowing NAS to automatically generate an optimal CNN architecture for the dataset in question [160].

White-box Automated side channel Attacks Recent works have explored the use of NAS for SCA initially defined white-box metrics to guide NAS, employing RANDOM and BAYESIAN search strategies[201] [201, 169]. These white-box evaluate the cumulative score of the secret key byte on a labeled attack dataset to measure the performance of the generated architecture [201]. This work was extended by introducing a reinforcement learning-based NAS



(a) ASCAD baseline [14] (b) ZAID baseline [206] (c) Our NAS base structure

Figure 6.1.: Baseline CNN architectures versus the NAS base structure

method, where the white-box objective functions as a reward for learning the Q-function, which then guides the search for the optimal architecture by selecting hyperparameters for the next architecture [169]. The Q-function uses metrics like guessing entropy and the number of traces (TST) to evaluate the performance of the system's secret key byte k^*. However, these approaches suffer from two key drawbacks. First, they use the attack dataset for hyperparameter optimization (HPO), leading to potential overfitting, which occurs when a model performs exceptionally well on the data it has seen but does not generalize to

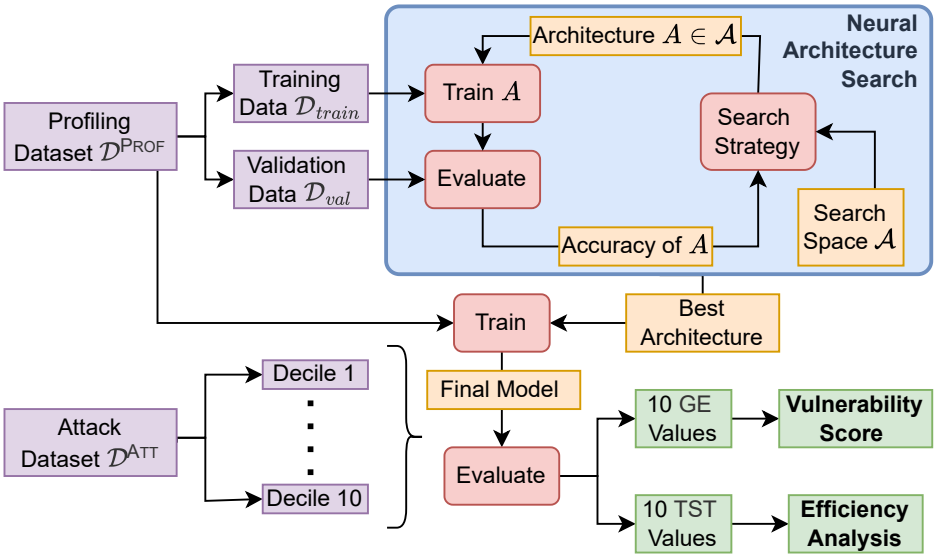


Figure 6.2.: Schematic of the NAS approach for black-box attacks

new, unseen data. To detect overfitting, a hold-out dataset must be withheld from model selection, parameter tuning, and training to evaluate the model's generalization ability. If the hold-out dataset is used in any part of the process, it can no longer assess generalization, leading to data-snooping issues [98]. This problem affects the experiments in [169] and [201], raising concerns about whether overfitting occurred.

Second, testing a model on the same dataset used to optimize its architecture overestimates its real-world performance. In scenarios where the architecture is optimized for an unknown key, this approach is unsuitable for a black-box setting, where the attack dataset remains unlabeled for training and is ultimately used to acquire the same secret key of the system. While this may be acceptable in white-box or gray-box settings, it is incompatible with the black-box assumptions of the work. Consequently, the performance results in [169] and [201] may not accurately represent real-world performance on unlabeled datasets and cannot be compared directly to the work.

6.1.2. Black-Box Automated Detection Approach

A black-box approach for HPO and neural architecture design, developed concurrently by Acharya et al. (2023) [1], is InfoNEAT. This method simultaneously evolves neural network architectures and hyperparameters, training a separate model for each key byte using a one-versus-rest (OVR) classification. Though its information-theoretic metrics suit hardware SCA, it cannot be directly compared to single-architecture methods, with the output layer predicting probability and real-valued scores for each possible key byte value [1]. Motivated by current automated solutions for SCAs, the goal is to develop an unbiased, optimal CNN architecture in a black-box setting using NAS. Figure 6.2 illustrates the NAS approach, adhering to the standard training-test-validation split commonly employed in ML methods to satisfy black-box requirements. The profiling dataset is divided into a training set $\mathcal{D}_{train}$, used for training the architecture, and a validation set $\mathcal{D}_{val}$, which evaluates its performance. The search strategy operates within a predefined search space $\mathcal{A}$ (c.f. Table 6.1), which contains various CNN architectures (1-D or 2-D based on the input). Initially, architectures $A \in \mathcal{A}$ are randomly selected (exploration), and their accuracy is used to inform future selections (exploitation). The training data $\mathcal{D}_{train}$ is used to train these architectures, and validation data $\mathcal{D}_{val}$ evaluates their performance. By only using the profiling dataset for both searching and evaluating architectures, the NAS method qualifies as a *black-box* approach [169], contrasting with the *white-box* methods used by Wu et al. (2024) [201] and Rijsdijk et al. (2021) [169], which rely on the test dataset, leading to biased performance estimates. After identifying the best-performing architecture, it is retrained on the full profiling dataset $\mathcal{D}_{\text{profiling}}$ to improve performance with minimal computational overhead. The attack dataset $\mathcal{D}_{attack}$ is divided into 10 equal parts (deciles), and each part is used to evaluate the VS using the GE values and efficiency using the TST values, as shown in Figure 6.2.

Vulnerability Score While the statistical tests are typically used to detect leakage, as outlined in Section 3.2.2, a more direct approach for IL detection is proposed employing the percentage of successful attacks through the VS. This method offers a more explicit confidence measure of IL based on the allowed number of guesses, i.e., GE to compromise a system.

To streamline the automated process of detection of leakage through NAS in AES-encrypted systems using the VS metric $\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*)$. This metric represents the ratio percentage of **successful attacks**, i.e., when the GE is less than or equal to 3 ($\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*) \leq 3$). As discussed in Section 2.3.3, the GE is approximated using R_a different attack datasets $\mathcal{D}_a^{\mathrm{ATT}}$ with N_a attack traces, and is computed using eq. (2.13).

The percentage of successful attacks, denoted by $\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*)$, is estimated by performing A_a different attacks with models generated by the black-box NAS approach with varying random seeds, calculated as

$$\mathrm{m}_{\mathrm{VS}}(\boldsymbol{r}^{N_a}, k^*) = \frac{1}{A_a} \sum_{i=1}^{A_a} [\![\mathrm{m}_{\mathrm{GE}}(\boldsymbol{r}^{N_a}, k^*) \leq 3]\!] \approxeq \frac{1}{A_a} \sum_{i=1}^{A_a} [\![\frac{1}{R_a} \sum_{a=1}^{R_a} \boldsymbol{r}^{N_a}[k^*] \leq 3]\!] \,. \tag{6.1}$$

This provides a precise measure of the system's vulnerability, used to assess the IL in AES encrypted systems, as discussed in Section 6.3.2. Moreover, TST values provide additional insights into the efficiency of a possible SCAs and how easily a given system can be compromised, as discussed in Section 6.3.3.

6.2. Experimental Setup

The performance of the final model returned by NAS largely depends on two key factors: the search strategy employed for NAS and the shape of the input features, as discussed in Section 6.2.1. First is the search strategy employed for NAS, which significantly influences search performance and runtime, and second is the shape of the input features. The methodology for this study is outlined in Section 6.1.1, while Section 6.2.2 presents the datasets used for the

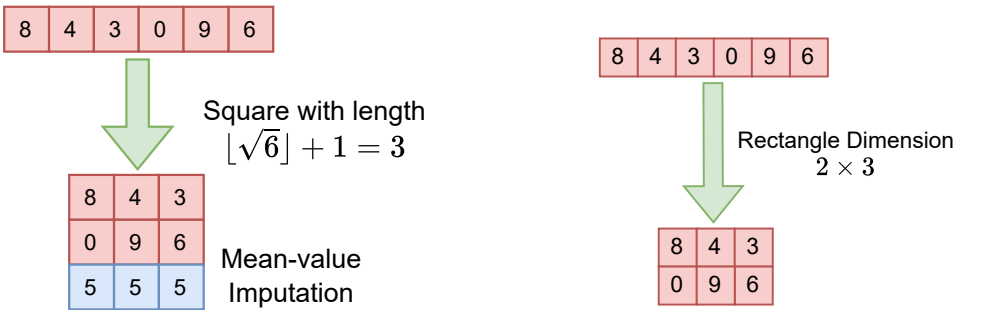
experiments. The goal of investigating the automated NAS setup in comparison to manually crafted fixed baseline CNN architectures, trained using hardware, is detailed in Appendix A.1.3.

6.2.1. NAS Parameters

The performance of the final model produced by NAS depends on two key factors: the search strategy, which affects the efficiency and runtime of the proposed NAS approach, and the input feature shape. The study aims to identify optimal choices for both.

Two-Dimensional Input Reshaping

The measurement traces of the datasets have to be transformed into the proper shape for the neural network to process. The most straightforward transformation is to produce a One-Dimensional input from the time series input and define a search space containing 1-D CNN architectures (c.f. Table 6.1) and to apply NAS to find an optimal architecture. While most of the 1-D CNN architectures that are explored for performing SCA to break AES-128 encryption were initially inspired by popular 2-D CNN architectures proposed for image classification, like VGGNet, Inceptionv3 [14], they are only applied on one-dimensional inputs. Recent work has shown that using 2-D CNNs could increase the accuracy and efficiency for breaking post-quantum key-exchange (PQKE) protocols [103, 87]. This also motivated me to explore using two-dimensional inputs for AES-128 attacks, applying NAS on a search space containing 2-D CNN architectures (c.f. Table 6.1). Two techniques are proposed to convert the one-dimensional input to two-dimensional input: Square and Rectangle. In the Square approach, a square of dimension $\lfloor\sqrt{d}\rfloor + 1$ is constructed, and the remaining parts filled with the mean value of the instance



(a) SQUARE Input Conversion (b) RECTANGLE Input Conversion

Figure 6.3.: Converting 1-D input to 2-D SQUARE and 2-D RECTANGLE input

are shown in Figure 6.3a. To form a RECTANGLE input, the dimensions closest to a square are determined using the two factors of d nearest to $\sqrt{d}$, avoiding the need for filling with imputed values, as illustrated in Figure 6.3b.

Search Strategies

The previous work on applying NAS for performing SCA only considered BAYESIAN and RANDOM search strategies [201], which are not necessarily time-efficient [160, 117]. The four search strategies RANDOM, GREEDY, HYPERBAND, and BAYESIAN were explored with 1000 fixed trials, utilizing their respective implementations provided by AutoKeras [100], as described in Section 2.2.4.

Architecture Search Space To perform NAS, the search space is defined based on the input shape of NAS. In Table 6.1, a search space is described containing various configurations for 2-D CNN architectures with RECTANGLE and SQUARE inputs, as well as a search space for 1-D CNN architectures with ONE-DIMENSIONAL inputs.

The hyperparameters for the convolutional layer are the kernel size and the number of filters, the pooling layer is the `pool-size` w_p, the number of strides and pooling operation type, and for a dense layer is the number of hidden units, as described in Section 2.2.4. The range for each hyperparameter is selected by

analyzing the related work [169, 201]. The search space includes all possible 1-D baseline CNN architectures [206, 14], ensuring that the search strategies could, in theory, find these fixed architectures and match their performance. In particular, RANDOM search strategy is guaranteed to identify these architectures eventually.

The range of each hyperparameter is also included from the search space designed for 1-D CNN architectures proposed by the current work on NAS for performing SCA [169, 201]. If the dataset's characteristics are known in advance, this search space can be reduced to improve search efficiency. However, the search space is not tailored to the dataset to ensure that the architecture design process remains fully automated, requiring no manual dataset analysis.

The network, layer, and dense block hyperparameters are the same for both 1-D CNN search space and 2-D CNN search space. Additionally, the range of convolutional kernel size, convolutional filters, and pooling types for each convolutional block is the same for both search spaces. To avoid the formation of invalid 2-D CNN architectures, the range of pooling strides is smaller, and the `pool-size` value is set using the kernel size, which is the suggested default of `AutoKeras` [100]. This reduces the search space for 2-D CNNs. The padding type is set to "same" for convolutional layers and "valid" for pooling layers in the 1-D CNNs search space. To prevent the creation of invalid 2-D CNNs, the padding type for both the convolutional and pooling layers is determined based on the kernel size of the convolutional layer, following the formulae proposed by `AutoKeras` [100]. The total number of possible hyperparameter configurations for 1-D CNNs is 40 758 681 600 and for 2-D CNNs is 2 264 371 200. For the experiments, the maximum number of trials is set to 1000 , implying that only 1000 possible architectures were explored by NAS out of such a large search space. Each search strategy is provided with the same budget limit, which means that only around 5×10^{-5} % of 2-D search space and 2.5×10^{-6} % of 1-D search space is explored to produce an optimal CNN architecture.

Table 6.1.: Overview of the Search Space for the NAS approach.

Hyperparameter Type	Hyperparameter	Possible Options
Whole Network	Optimizer	{`Adam`, `Adam_with_weight_decay` }
	Learning rate	{1e1, 5e2, 1e2, 5e3, 1e3, 5e4, 1e4, 5e5, 1e5}
Every Layer	Dropout	$\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$
	Use Batch Normalization	{True, False}
	Activation Function	{ `ReLU`, `SELU`, `Elu`, `Tanh` }
Convolutional Block	# Blocks	$\{1, 2, 3, 4, 5\}$
	Convolutional Kernel Size	{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}
	Convolutional Filters	{2, 8, 16, 32, 64, 128, 256}
	Pooling Type	{'max' , 'average'}
	Pooling Strides 1-D CNN	{2, 3, 4, 5, 6, 7, 8, 9, 10}
	Pooling Poolsize 1-D CNN	{2, 3, 4, 5}
	Pooling Strides 2-D CNN	{2, 4}
	Pooling Poolsize 2-D CNN	Convolutional Kernel Size-1
Dense Block	# Blocks	{1, 2, 3}
	Hidden Units	{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}

6.2.2. Dataset Description

To investigate the behavior of NAS across a wide range of settings and enable direct comparisons with other works in hardware SCAs, five well-known datasets were selected: ASCAD v1, DPA contest v4.1, AES_RD, AES_HD, and CHES CTF 2018. These datasets have been extensively studied in prior research and are recorded from an AES-128 encrypted system. These datasets record implementations of AES-128, meaning the same overall approach should work automatically for each without needing dataset-specific tuning. However, the systems incorporate different ILs countermeasures, e.g., masking or random delays. The attack setup (dataset, target byte, features) is identical to the one described in [206, 14], with the configurations for all these datasets listed in Table 6.2.

ASCAD The ASCAD dataset was proposed as a benchmark dataset, containing expectation-maximization (EM) radiation measurements obtained from an ATMega8515 (8-bit micro-controller with AVR architecture) device running a masked implementation of AES-128 [14].

Table 6.2.: Details of the datasets acquired from the AES-encrypted systems.

Dataset name	# Features	# Profiling traces	# Attack traces	Attack byte
ASCAD_f	700	50000	10000	2
ASCAD_f desync50	700	50000	10000	2
ASCAD_f desync100	700	50000	10000	2
ASCAD_r	1400	50000	100000	2
ASCAD_r desync50	1400	50000	100000	2
ASCAD_r desync100	1400	50000	100000	2
CHES CTF	2200	45000	5000	2
AES_HD	1250	50000	25000	0
AES_RD	3500	25000	25000	0
DP4CONTEST	4000	4500	5000	0

This dataset is available in different versions, and the original v1 dataset is utilized in both with the fixed secret key (ASCAD_f[1]) and variable secret key (ASCAD_r[2]) variants. The traces in these datasets have been aligned such that the AES computation always starts at the same sample within each trace. The points of interest in the raw data have been analyzed using the signal-to-noise ratio, and only a small subset of samples from the full traces is used [14]. Additionally, versions that add random amounts of desynchronization to each trace are considered. These versions were created by Benadjila et al. (2020) [14] to simulate imprecise temporal alignment of the traces by adding an artificial jitter to each trace. The traces have been desynchronized by a maximum of 50 and 100 samples, resulting in the 6 different varieties of the ASCAD dataset listed in Table 6.2.

[1] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key/

[2] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key/

CHES CTF The CHES CTF dataset comprises traces initially produced for the CHES 2018 AES-128 CTF challenge. Note that the dataset used in this study [3] is a single reduced version (45 000 traces) derived from the original measurements (500 000 traces) and not identical to the datasets published as part of the actual contest[4], which consists of 6 different sets (42 000 traces). This reduced version is used in the AISY framework and has already been preprocessed [144].

AES_RD AES_RD[5] was initially used to investigate random delay countermeasures [37]. The traces for this dataset are collected from an 8-bit ATMEL AVR microcontroller running an AES-128 implementation incorporating random delays. The converted dataset as analyzed in Zaid et al. (2019) [206] is used[6].

AES_HD AES_HD[7] dataset contains EM measurements obtained from Xilinx Virtex-5 FPGA (coded in VHDL) implementing an unprotected AES-128 implementation [148]. A big difference in this dataset is that it records the AES decryption operation instead of the encryption operation. The labels are generated for a difference leakage model based on the ciphertext bytes c_i^j used in the decryption, specifically the 12th (c_i^{11}) and 8th (c_i^7) ciphertext bytes. The resulting label is then calculated with $\phi(c_i, k_i) = sbox^{-1}(c_i^{11} \oplus k_i) \oplus c_i^7$ (c.f. section 2.3.3). Once again, the converted dataset as analyzed in Zaid et al. (2019) [206] is used[8].

[3] http://aisylabdatasets.ewi.tudelft.nl/ches_ctf.h5
[4] https://chesctf.riscure.com/2018/content?show=training
[5] https://github.com/ikizhvatov/randomdelays-traces/
[6] https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA/tree/master/AES_RD/AES_RD_dataset
[7] https://github.com/AESHD/AES_HD_Dataset/
[8] https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA/blob/master/AES_HD/AES_HD_dataset.zip

DPAv4 The DPAv4 dataset[9] contains traces obtained from ATMEL AVR-163 microcontroller running an AES-128 implementation protected with rotating Sbox masking (RSM) [16]. It was used in the fourth version of the DPA contest, from which only the "improved" masked AES-128 target contained in dataset version 4.2 is employed. Again the extracted dataset[10] from [206] is actually used. To be consistent while comparing the approach with the performance of the baselines proposed by [206], the mask value is assumed to be known, essentially nullifying the masking.

6.3. Parameter Study Results

The complete parameter study outlined in Section 6.2 combined the possible options for search strategy and input shape. These configurations were applied to the 10 datasets detailed in Section 6.2.2 for an identity (ID) leakage model. Additionally, the baseline architectures described in Appendix A.1.3 were trained for each dataset.

6.3.1. Optimal Parameters

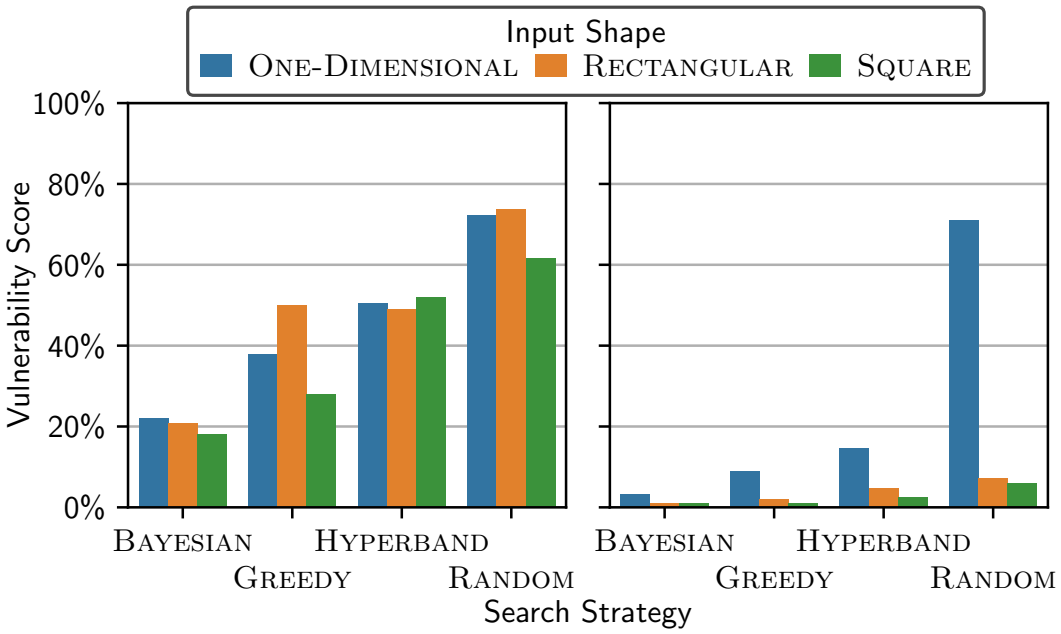
To address the second question, the VS is plotted for every possible NAS parameter combination of input shape and search strategy. These plots were created separately for synchronized and desynchronized datasets due to the significant difference in performance between the two.

[9] http://www.dpacontest.org/v4/42_traces.php/

[10] https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA/blob/master/DPA-contest%20v4/DPAv4_dataset.zip

Synchronized Datasets

Figure 6.4a shows the influence of the choice of each possible NAS parameter combination (input shape and the search strategy) on the overall performance of synchronized datasets for ID leakage. This clearly shows that the choice of search strategy has a significant impact on the VS, which rises from around 20 % to around 70 % when going from Bayesian search via Greedy and Hyperband to Random search. Random search strategy outperforms the other strategies. Still, it comes at the price of being slower, taking about 5 times longer than Greedy and Hyperband (c.f. Appendix A.1.3), mainly because it keeps exploring the search space until the limit of 1000 trials is reached. For synchronized datasets, using the Hyperband strategy might be a viable alternative, as it still produces a CNN with a VS of around 50 % while being substantially faster than Random search. When comparing input



(a) Synchronized datasets (b) Desynchronized datasets

Figure 6.4.: Influence of NAS parameters on detecting ILs in AES-encrypted systems

shapes, there does not appear to be a consistent difference between 1-D and 2-D

for synchronized datasets, demonstrating the ability of NAS to adjust to vastly different situations. This demonstrates that 2-D CNNs offer no advantage over the 1-D input shape.

Desynchronized Datasets

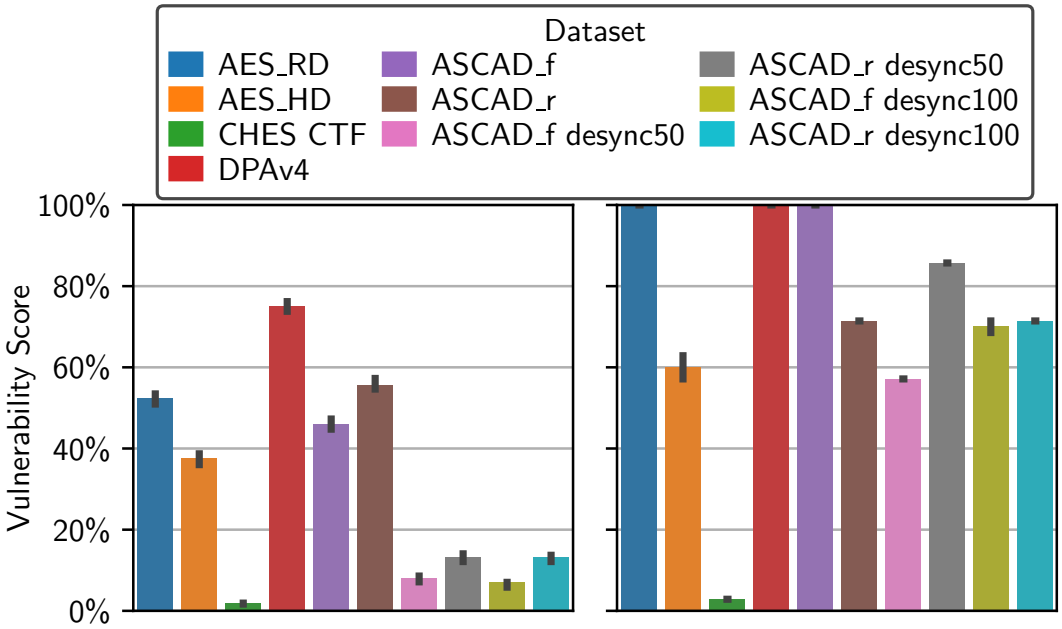
Figure 6.4b shows the influence of the parameter choices on the VS for desynchronized datasets. The difference between RANDOM and its competitors grows even larger on these datasets, so choosing HYPERBAND for its speed is no longer a viable option. When desynchronization is introduced, the input shape plays a major role. The 1-D CNN architecture can compensate for desynchronization to a large degree, with its performance essentially unchanged when paired with RANDOM search. Wouters et al. (2020) [200] observed a similar behavior where the first convolutional block successfully removes the desynchronization in the dataset by using the convolutional and pooling operation on neighboring values in the trace. Converting 1-D to 2-D inputs changes the local relationship between neighboring values, which makes re-synchronizing the traces more difficult. The 2-D architectures struggle with re-synchronization in the experiments and are not viable for desynchronized datasets.

6.3.2. Parameter Reliability

To determine the overall reliability of the NAS approach, the VS is determined for all experiments executed on each dataset in the ID leakage model. In this context, an attack is considered successful if the final GE reaches a value of 1 after processing all traces in the respective attack dataset decile. The per-dataset attack VS is shown in Figure 6.5a, providing a rough indication of the difficulty level of attacking each dataset. One notable observation is the relative ease of attacking the DPAv4 dataset: Even when accounting for all suboptimal

combinations of the search strategy and input shape, over 75 % of the attacks were successful. This is unsurprising, as the considered variant of the dataset effectively contains no countermeasures, as detailed in Section 6.2.2.

When comparing the synchronized versions of ASCAD_f and ASCAD_r to their desynchronized counterparts, the degradation in reliability caused by the increased difficulty incurred by desynchronization is evident. However, some of the attacks were still successful. The disastrous performance of NAS on the CHES CTF dataset must also be highlighted, where only a handful of attacks successfully recovered the complete identity value. This can be traced back to a reduced number of 500 attack traces available for the attack because of the decile split, with the convergence indicating that all models would have succeeded given a larger attack dataset.



(a) Over all parameter choices (b) Using 1-D and RANDOM

Figure 6.5.: Vulnerability score for various AES-encrypted systems

Reliability of Optimal Parameters

The analysis above demonstrates that the highest likelihood of producing a CNN model capable of successfully breaking the system is created by combining the RANDOM search strategy with ONE-DIMENSIONAL inputs, particularly in desynchronized scenarios. To evaluate the VS achieved with this specific combination, it is plotted per dataset in Figure 6.5b. The results indicate that, except for the CHES CTF dataset, over 57 % of the attacks were successful, with some datasets achieving a remarkable VS of 100 %. The anomaly observed in AES_HD and CHES CTF will be investigated further. Thus, it can be concluded that the combination of a ONE-DIMENSIONAL input shape and the RANDOM search strategy is the optimal choice for reliably creating successful attack models.

6.3.3. Efficiency Analysis

As concluded, using the RANDOM search strategy and ONE-DIMENSIONAL input shape yields CNN architectures capable of performing SCA with a high VS. Additionally, the efficiency of these NAS architectures and their comparison to traditional fixed architectures were also investigated.

Comparison of Trace Sufficiency Threshold For a fair comparison, the baseline architectures presented in Appendix A.1.3 were considered in this work. Efficiency is measured using the TST, which quantifies the number of attack traces required for the model to achieve a GE of 1, with lower TST values indicating greater efficiency. Table 6.3 shows the median TST values for the 7 NAS models generated by the RANDOM search strategy on ONE-DIMENSIONAL inputs, compared against the ASCAD and ZAID baseline architectures.

On the simpler datasets such as AES_RD and DPAv4, the NAS models perform instantaneous attacks, requiring only a single trace. For 4 out of 10 datasets, the NAS models outperform the baselines. However, for desynchronized datasets,

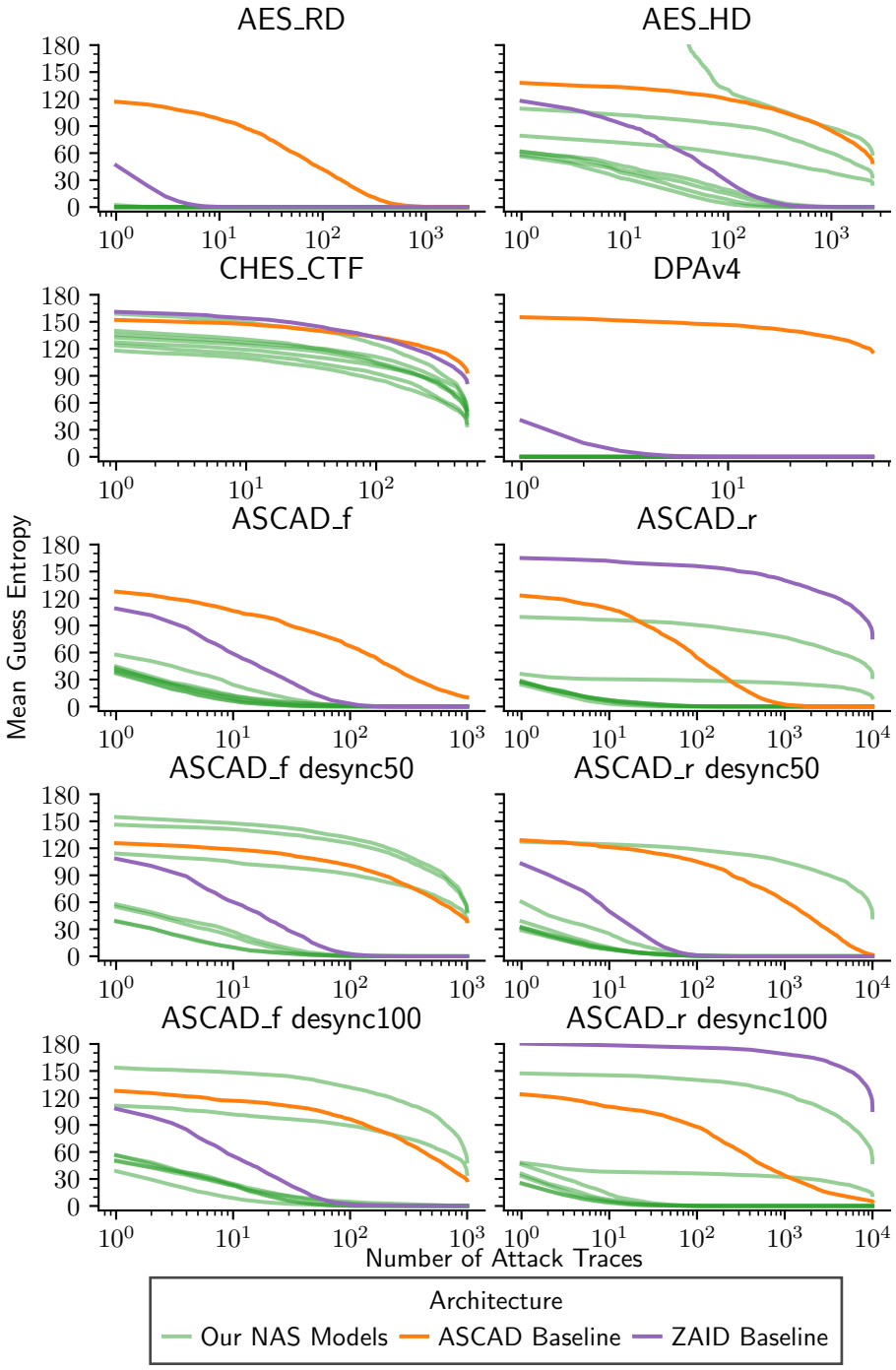


Figure 6.6.: Convergence of the 7 NAS models compared to the fixed baseline architectures for each dataset

Table 6.3.: Median TST comparison across datasets, italics indicate GE not reaching 1; the best model per dataset is in bold.

Dataset	ASCAD	ZAID	Our NAS
AES_RD	270.3	3.0	**1.0**
AES_HD	*2500.0*	**504.4**	1172.8
CHES CTF	**498.5**	*500.0*	*500.0*
DPAv4	49.3	2.9	**1.0**
ASCAD_f 0ms	703.5	**116.5**	118.3
ASCAD_r 0ms	322.0	*10000.0*	**136.3**
ASCAD_f 50ms	920.9	**136.9**	202.0
ASCAD_r 50ms	4449.0	**56.1**	139.7
ASCAD_f 100ms	974.6	**82.5**	391.7
ASCAD_r 100ms	6664.2	*10000.0*	**87.9**

especially where the ZAID baseline is specialized, NAS only surpasses the ASCAD baseline and the Notably, baselines fail to produce successful attacks for CHES CTF, synchronized ASCAD, and ASCAD_f desync50 datasets. This highlights an issue when relying on TST alone, as unsuccessful attacks are still counted in the total number of attack traces, influencing the average TST. Figure 6.6 further illustrates the GE convergence of the 7 NAS models compared to the fixed architectures for each dataset, which shows the convergence of each NAS model and only takes the average GE over the 10 attack deciles. In most cases, the best NAS model outperforms the baselines in 6 out of 10 datasets. However, individual outliers exhibit significantly slower convergence, particularly in the AES_HD and ASCAD_r datasets. For example, while most NAS architectures perform comparably to the ZAID baseline on AES_HD, 3 out of 7 models become outliers with slower convergence, with one model diverging before achieving a decent GE toward the end. A similar pattern is seen in ASCAD_r, where 5 out of 7 NAS models surpass the ASCAD baseline, but 2 models show much slower convergence. Outliers appear in 6 out of 10 cases on desynchronized datasets. Interestingly, even the fixed ZAID architecture for

ASCAD_f desync50 is affected by an outlier model, highlighting that random variations can impact both fixed and NAS models. Such variability underscores that while NAS often matches or exceeds the performance of fixed architectures, its generalization capability can sometimes falter due to randomization and slight changes in the dataset splits.

6.3.4. Summary

The parameter study highlights the performance and efficiency of NAS-generated architectures compared to traditional fixed baselines for SCA on various datasets. For synchronized datasets, RANDOM search with 1-D input shapes proved to be the optimal parameter combination, delivering consistent and high VS. At the same time, HYPERBAND provided a faster alternative with moderate performance. The performance gap widened on desynchronized datasets, with RANDOM search being the only viable strategy, as 2-D input shapes struggled to handle desynchronization effectively.

Generalization NAS achieved state-of-the-art performance on several datasets exhibiting significant variability based on the train-validation splits, indicating sensitivity to small dataset changes. For instance, the best NAS models on ASCAD_r and AES_HD datasets were highly efficient in some splits but underperformed in others, highlighting the generalization challenge. This variability suggests that while NAS can find optimal architectures for specific datasets, its generalization across different splits or datasets remains a concern, as it may overfit specific patterns in the training data. Such overfitting can reduce robustness, particularly in side-channel analysis tasks where slight changes in attack scenarios or desynchronization levels can drastically affect model performance. This issue is common in ML processes where randomization or small changes in data splits can lead to varying results. Additionally, in the CHES_CTF dataset, splitting into deciles revealed that none of the models achieved a GE of 1 by the end of the experiment, possibly due to the small

dataset size or the complexity of the attack traces. These findings suggest that while NAS models can match or outperform fixed architectures in terms of attack efficiency, their inconsistent performance remains a limitation, and generalization becomes a critical question when using NAS for full key recovery, as described in Section 2.3.3. A future research direction could investigate using techniques such as cross-validation or regularization to reduce overfitting, helping ensure that NAS architectures perform consistently across a broader range of datasets and scenarios. Despite this variability, NAS can potentially eliminate manual per-dataset architecture design, but further refinement is needed to ensure consistent performance across a wide range of datasets.

7. Summary and Future Directions

The final chapter provides an overall conclusion of the thesis, summarizing key findings and contributions in Section 7.1. Future research directions are outlined in Section 7.2, emphasizing improving ILD frameworks and expanding their applications to other cryptographic systems and real-world security challenges. The conclusion and outlook are partially published in Gupta et al. (2024) [81] and Gupta et al. (2023) [79].

7.1. Conclusion

This thesis introduces a comprehensive theoretical framework for addressing the ILD problem in cryptographic systems, bridging gaps between the channel capacity from information theory and statistical learning to justify the reasons for widespread usage and accuracy of ML, specifically DL methods. A central contribution of this work is the proposal of the leakage assessment score (LAS) as a generalized measure of IL in systems, which is derived using the connection between MI and Bayes predictor and calculated by assessing the Bayes predictor performance. This metric is particularly important since when MI estimation becomes increasingly challenging with 256 possible classes, representing the secret key byte, detecting ILs in AES-encrypted systems benefits from the proposed VS metric, which is based on evaluating LAS using the GE (key byte rank) of the system. I establish robust methodologies for identifying

vulnerabilities by detecting leakages in systems using different variants of LAS and enhancing cybersecurity through this systematic framework. Due to the learning consistency of DL-based and ensemble ML methods using bagging and boosting like random forest classifier (RF), gradient boosting machine (GBM), I proposed using the state-of-the-art automated machine learning (AutoML) tools, such as TabPFN and AutoGluon to induce the Bayes predictor.

I introduced two techniques to quantify and detect IL by estimating MI between the observable and secret information, leveraging the approximation of Bayes predictor performance in terms of Log-Loss and accuracy. First, I compared the proposed MI estimation approaches with the state-of-the-art by performing the experiments on the systems simulated using multivariate normal (MVN) generating classification datasets with known MI. Several approaches are proposed to detect ILs in the systems using statistical tests, like the cut-off technique using one-sample t-test (OTT) on the MI estimates, Fisher's exact test (FET) on the confusion matrices (CMs), and paired t-test (PTT) on accuracy estimates of the AutoML tools. These tests are performed multiple times, and their results are aggregated using the Holm-Bonferroni correction to ensure reliable and confident decisions regarding the presence of IL in the cryptographic systems, making it robust towards noise. I also proposed using the black-box NAS (using only profiling dataset) as a promising realist approach for performing SCAs on AES-encrypted systems exploiting the power consumption or EMRs to reveal the secret key, extending its application to perform vulnerability analysis of the systems using the VS metric (using GE). Through a large-scale parameter study, I assess the impact of NAS configurations, focusing on search strategies and input shapes (1-D and 2-D), on the performance of CNN architectures. I evaluate four search strategies implemented by `AutoKeras`—Random, Greedy, Hyperband, and Bayesian —and explore the transformation of one-dimensional inputs into two-dimensional rectangular and square formats to enable the use of 2-D CNNs, inspired by VGGNet and Inceptionv3 proposed for image classification. This comprehensive study examines these factors across 10 datasets in the ID leakage model, providing insights into their influence on model performance. Additionally, I provided a detailed analysis of the

efficiency of the proposed automated NAS approach by analyzing the TST by performing multiple attacks using best models generated across 7 independent NAS runs to account for the uncertainty in estimates. This analysis offers a detailed perspective on the ease with which the AES-encrypted system can be compromised.

Our proposed approaches, employing two powerful AutoML tools, enable automated, scalable, and precise approximation of Bayes predictor to estimate LAS, also MI effectively. This, in turn, facilitates the detection of IL in imbalanced, high-dimensional datasets, significantly reducing reliance on manual configurations and traditional statistical techniques. Our empirical findings demonstrate that the proposed approach, which approximates calibrated Log-Loss using TabPFN, is highly effective and robust in precisely estimating MI in synthetic datasets and detecting side channel leaks via network traces in OpenSSL Transport Layer Security (TLS) servers compared to state-of-the-art methods, as detailed in Chapters 4 and 5. Furthermore, this work concludes that the choice and requirement of calibration techniques for Log-Loss estimation depend on the characteristics of the system datasets. Upon detailed analysis of the results in analyzing the vulnerability of the AES-encrypted systems, I concluded that using the Random search strategy on one-dimensional inputs yields the best-performing CNN architectures for the available medium-sized computational budgets. I compared the efficiency of the NAS models with state-of-the-art CNN baselines and demonstrated that, for the synchronized datasets generated by systems implementing no or weak countermeasures, the 7 best models generated by NAS were more efficient, requiring fewer traces (low TST) than the baselines.

Considering that the proposed approach matched the performance of hand-crafted architectures, NAS demonstrates its potential for fully automated attacks on devices or datasets with unknown characteristics. These experiments underscore the importance of exploration in selecting search strategies for hardware attacks, emphasizing the need to account for broader parameter spaces. Real-world attackers with access to larger budgets could achieve even

better results using RANDOM search, positioning the presented findings as a conservative estimate of actual capabilities. Additionally, the flexibility of NAS enables unbiased comparisons of SCA methods, addressing the limitations of current evaluations that rely on fixed architectures, such as predefined loss functions. A notable challenge observed was the susceptibility of ML models to variations in training datasets—a factor often overlooked in existing ML-based SCA studies. This issue became apparent through repeated training with different validation splits, highlighting the need for a systematic discussion on improving consistency in performance evaluations for ML-based SCA.

In summary, the empirical evaluation showcases the robustness of the proposed approaches across diverse datasets, including leakage via processing time and error codes in the network traces of the OpenSSL TLS servers and hardware side channels in AES-encrypted systems. This thesis establishes a foundation for an automated ILD adaptable to various real-world cybersecurity scenarios by addressing key challenges such as noise, imbalance, and generalizability. This work also emphasizes the importance of unbiased and consistent performance evaluation in ML-based SCA research, advocating for broader discussions on mitigating variability in training datasets and improving reliability in evaluation protocols. The proposed methodologies advance the state-of-the-art in ILD and pave the way for fully automated and efficient side channel detection in cryptographic systems. In conclusion, this thesis contributes to the theoretical and practical aspects of ILD, offering scalable and adaptable solutions that address the evolving challenges of cybersecurity in cryptographic environments.

7.2. Future Work

In the future, I would like to explore extending the current framework to estimate Rényi entropy, leveraging its tunable parameter α to analyze leakage distributions more comprehensively [57, 13]. This approach would provide deeper insights into how uncertainty and adversarial success probabilities vary across different leakage scenarios, complementing MI-based evaluations [155].

I also want to provide a fully automated solution on the high dimensional dataset with numerous classes, mitigating the limitations of the approaches using TabPFN [89]. Furthermore, this work concludes that the choice and requirement of calibration techniques for Log-Loss estimation depend on the characteristics of the system datasets. Specifically, I intend to develop an end-to-end AutoML tool, containing possible usage of MLPs and CNNs as well, that selects appropriate dimensionality reduction techniques in conjunction with calibration methods to accurately estimate MI and detect IL in any given cryptographic system. Considering that the Mid-Point approach does not account for system dataset imbalance, it is crucial to determine a Bayes predictor that reduces metrics like balanced error-rate (BER) or maximizes mathews correlation coefficient (MCC), F1-score, and so on, as defined in Section 2.2.2. To improve the estimation using the Mid-Point approach, I intend to expand upon the relationship between BER and MI for binary classification provided by Zhao et al. (2013) [210] and adapt it to accommodate multiple classes. Another future direction could be to integrate the fast Westfall–Young permutation procedure to improve statistical power and empirically adjust the Holm-Bonferroni cut-off thresholds to a widely accepted significance level of $\alpha = 0.05$ [186, 46]. Future work could also involve evaluating the applicability and limitations of classical and neural MI estimation methods in more complex scenarios, such as high-dimensional settings, sparse interactions, long-tailed distributions, and high MI values, beyond the synthetic MVN datasets analyzed [42]. Additionally, future work includes detecting leaks through other side channels such as CPU caches, speech recognition, Soundcomber, and WiFi signal radiations [137, 150], as well as extending these methods to other cryptographic protocols. It also involves exploring advanced NAS strategies to enhance further the adaptability and efficiency of automated SCAs and ILD systems. I also intend to explore adaptive techniques that dynamically adjust and fine-tune the IL detection process based on changing environments or evolving attack strategies, utilizing reinforcement learning, online learning, and other adaptive methods [54].

While most of the current research in detecting hardware ILs in side-information observable via the power consumption or EMR side channels of the AES-encrypted systems primarily addresses systems employing more straightforward defenses, such as Boolean masking. Our study was restricted to detecting the ID leakage in AES-encrypted systems. At the same time, I intend also to analyze the vulnerability in terms of the hamming weight (HW) or hamming distance (HD) leakage. Future work could explore detecting hardware ILs in systems incorporating advanced countermeasures like affine masking to offer more robust resistance to SCAs in datasets like ASCADv2. Additionally, analyzing complete input traces instead of truncated ones could provide deeper insights into the vulnerabilities of systems employing higher-order masking and complex defenses. I focused on four search strategies, but more advanced alternatives have proven to be more time-efficient and effective than, for example, search in finding optimal architectures [160]. Another potential efficiency enhancement is early stopping, which involves aborting hyperparameter optimization runs early if performance is consistently poor or no further improvement is observed.

8. Bibliography

[1] Rabin Yu Acharya, Fatemeh Ganji, and Domenic Forte. "Information Theory-based Evolution of Neural Networks for Side-channel Analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023.1 (Nov. 2023), pp. 401–437. DOI: 10.46586/tches.v2023.i1.401-437. URL: https://doi.org/10.46586/tches.v2023.i1.401-437.

[2] Onur Acıiçmez and Çetin Kaya Koç. "Trace-Driven Cache Attacks on AES (Short Paper)". In: *Information and Communications Security*. Ed. by Peng Ning, Sihan Qing, and Ninghui Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 112–121. ISBN: 978-3-540-49497-3.

[3] Md. Abdullah Al Alamin and Gias Uddin. "How far are we with automated machine learning? Characterization and Challenges of AutoML Toolkits". In: *Empirical Software Engineering* 29.4 (2024), p. 91. DOI: 10.1007/s10664-024-10450-y. URL: https://doi.org/10.1007/s10664-024-10450-y.

[4] U. Ali and O. Khan. "MultiCon: An Efficient Timing-based Side Channel Attack on Shared Memory Multicores". In: *2022 IEEE 40th International Conference on Computer Design (ICCD)*. 2022 IEEE 40th International Conference on Computer Design (ICCD). 2022, pp. 97–104. DOI: 10.1109/ICCD56317.2022.00024. URL: https://doi.org/10.1109/ICCD56317.2022.00024.

[5] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. "DROWN: Breaking TLS Using SSLv2". In: *25th USENIX Security Symposium (USENIX Security 16)*. Ed. by Thorsten Holz and Stefan

Savage. Austin, TX, USA: USENIX Association, Aug. 2016, pp. 689–706. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_aviram.pdf.

[6] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. "Efficient Padding Oracle Attacks on Cryptographic Hardware". In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2012, pp. 608–625. DOI: 10.1007/978-3-642-32009-5_36.

[7] R. E. Barlow and H. D. Brunk. "The Isotonic Regression Problem and Its Dual". In: *Journal of the American Statistical Association* 67.337 (Nov. 1972), pp. 140–147. ISSN: 01621459. DOI: 10.2307/2284712.

[8] Vanessa Barros and Jérôme Rousseau. "Shortest Distance Between Multiple Orbits and Generalized Fractal Dimensions". In: *Annales Henri Poincaré* 22.6 (June 2021), pp. 1853–1885. ISSN: 1424-0661. DOI: 10.1007/s00023-021-01039-y.

[9] Omid Bazangani, Alexandre Iooss, Ileana Buhan, and Lejla Batina. "ABBY: Automating Leakage Modelling for Side-Channel Analysis". In: *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (ASIA CCS '24)*. Singapore, Singapore: Association for Computing Machinery, Apr. 2024, pp. 231–244. DOI: 10.1145/3634737.3637665. URL: https://doi.org/10.1145/3634737.3637665.

[10] Julien Béguinot, Wei Cheng, Sylvain Guilley, and Olivier Rioul. "Be My Guess: Guessing Entropy vs. Success Rate for Evaluating Side-Channel Attacks of Secure Chips". In: *2022 25th Euromicro Conference on Digital System Design (DSD)*. Maspalomas, Spain, Sept. 2022, pp. 496–503. DOI: 10.1109/DSD57027.2022.00072.

[11] Julien Béguinot and Olivier Rioul. "What can Information Guess? Guessing Advantage vs. Rényi Entropy for Small Leakages". In: *2024 IEEE International Symposium on Information Theory (ISIT)*. 2024 IEEE International

Symposium on Information Theory (ISIT). 2024, pp. 2963–2968. DOI: 10.1109/ISIT57864.2024.10619150. URL: https://doi.org/10.1109/ISIT57864.2024.10619150.

[12] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. "Mutual Information Neural Estimation". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Stockholmsmässan, Stockholm, Sweden: Proceedings of Machine Learning Research, July 2018, pp. 531–540.

[13] M. Ben-Bassat and J. Raviv. "Renyi's Entropy and the Probability of Error". In: *IEEE Transactions on Information Theory* 24.3 (May 1978), pp. 324–331. ISSN: 0018-9448. DOI: 10.1109/TIT.1978.1055890. URL: https://doi.org/10.1109/TIT.1978.1055890.

[14] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. "Deep Learning for Side-Channel Analysis and Introduction to ASCAD Database". In: *Journal of Cryptographic Engineering* 10.2 (June 2020), pp. 163–188. ISSN: 2190-8516. DOI: 10.1007/s13389-019-00220-8. URL: https://doi.org/10.1007/s13389-019-00220-8.

[15] Anastasija Berlinblau. "Detection of Timing Side Channels: Extending the AutoSCA Tool". Bachelor's Thesis. Bergische Universität Wuppertal, 2021.

[16] Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. "Analysis and Improvements of the DPA Contest v4 Implementation". In: *Security, Privacy, and Applied Cryptography Engineering – 4th International Conference, SPACE 2014*. Ed. by Rajat Subhra Chakraborty, Vashek Matyas, and Patrick Schaumont. Vol. 8804. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 201–218. ISBN: 978-3-319-12060-7. DOI: 10.1007/978-3-319-12060-7_14. URL: https://doi.org/10.1007/978-3-319-12060-7_14.

[17] Bhaskar Bhattacharya and Desale Habtzghi. "Median of thepValue Under the Alternative Hypothesis". In: *The American Statistician* 56.3 (Nov. 2002), pp. 202–206. ISSN: 0003-1305, 1537-2731. DOI: 10.1198/000313002146.

[18] Gérard Biau, Luc Devroye, and Gábor Lugosi. "Consistency of Random Forests and Other Averaging Classifiers". In: *Journal of Machine Learning Research* 9.66 (June 2008), pp. 2015–2033. ISSN: 1532-4435.

[19] Christopher M. Bishop. *Probability Distributions*. New York, NY: Springer New York, NY, 2006. ISBN: 978-0387310732. DOI: 10.5555/1162264.

[20] Daniel Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *Advances in Cryptology – CRYPTO '98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1998, pp. 1–12. ISBN: 978-3-540-68462-6. DOI: 10.1007/BFb0055716.

[21] Hanno Böck, Juraj Somorovsky, and Craig Young. "Return Of Bleichenbacher's Oracle Threat (ROBOT)". In: *27th USENIX Security Symposium (USENIX Security 18)*. Ed. by William Enck and Adrienne Porter Felt. Baltimore, MD, USA: USENIX Association, Aug. 2018, pp. 817–849. ISBN: 978-1-939133-05-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/boeck.

[22] GLENN W. BRIER. "VERIFICATION OF FORECASTS EXPRESSED IN TERMS OF PROBABILITY". In: *Monthly Weather Review* 78.1 (Jan. 1950), pp. 1–3. DOI: 10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/mwre/78/1/1520-0493_1950_078_0001_vofeit_2_0_co_2.xml.

[23] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. "Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing". In: *Cryptographic Hardware and Embedded Systems - CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*. Ed. by Wieland Fischer and Naofumi Homma. Vol. 10529. Lecture Notes in Computer Science. Springer International Publishing. Berlin, Heidelberg: Springer, Sept. 2017, pp. 45–68. ISBN: 978-3-319-66787-4. DOI: 10.1007/978-3-319-66787-4_3. URL: https://doi.org/10.1007/978-3-319-66787-4_3.

[24] Gregory Camilli. "The Relationship between Fisher's Exact Test and Pearson's Chi-Square Test: A Bayesian Perspective". In: *Psychometrika* 60.2 (June 1995), pp. 305–312. ISSN: 1860-0980. DOI: 10.1007/BF02301418. URL: https://doi.org/10.1007/BF02301418.

[25] Lipeng Chang, Yuechuan Wei, Shuiyu He, and Xiaozhong Pan. "Research on Side-Channel Analysis Based on Deep Learning with Different Sample Data". In: *Applied Sciences* 12.16 (Aug. 2022). ISSN: 2076-3417. DOI: 10.3390/app12168246. URL: https://www.mdpi.com/2076-3417/12/16/8246.

[26] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. "Towards Sound Approaches to Counteract Power-Analysis Attacks". In: *Advances in Cryptology — CRYPTO '99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 1999, pp. 398–412. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_26.

[27] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. "Template Attacks". In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Redwood Shores, CA, USA: Springer Berlin Heidelberg, Aug. 2003, pp. 13–28. ISBN: 978-3-540-36400-9. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5_3.

[28] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. "Statistical Measurement of Information Leakage". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Javier Esparza and Rupak Majumdar. Vol. 6015. Lecture Notes in Computer Science. Paphos, Cyprus: Springer Berlin Heidelberg, 2010, pp. 390–404. ISBN: 978-3-642-12002-2. DOI: 10.1007/978-3-642-12002-2_33. URL: https://doi.org/10.1007/978-3-642-12002-2_33.

[29] Boru Chen, Yingchen Wang, Pradyumna Shome, Christopher Fletcher, David Kohlbrenner, Riccardo Paccagnella, and Daniel Genkin. "GoFetch: Breaking Constant-Time Cryptographic Implementations Using Data Memory-

Dependent Prefetchers". In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1117–1134. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/chen-boru.

[30] Weiwei Cheng and Eyke Hüllermeier. "Probability Estimation for Multi-class Classification Based on Label Ranking". In: *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 83–98. ISBN: 978-3-642-33486-3.

[31] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. "Best Information is Most Successful: Mutual Information and Success Rate in Side-Channel Analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.2 (Feb. 2019), pp. 49–79. ISSN: 2569-2925. DOI: 10.13154/tches.v2019.i2.49-79. URL: https://tches.iacr.org/index.php/TCHES/article/view/7385.

[32] Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. "The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation". In: *BioData Mining* 14.1 (Feb. 2021), p. 13. ISSN: 1756-0381. DOI: 10.1186/s13040-021-00244-z.

[33] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[34] Tom Chothia and Apratim Guha. "A Statistical Test for Information Leaks Using Continuous Mutual Information". In: *2011 IEEE 24th Computer Security Foundations Symposium*. 2011, pp. 177–190. DOI: 10.1109/CSF.2011.19.

[35] Aakash Chowdhury, Carlo Brunetta, Arnab Roy, and Elisabeth Oswald. *Leakage Certification Made Simple*. Cryptology ePrint Archive, Paper 2022/1201. 2022. URL: https://eprint.iacr.org/2022/1201.

[36] Wikipedia contributors. *Rényi Entropy*. 2023. URL: https://en.wikipedia.org/wiki/R%5C'enyi_entropy.

[37] Jean-Sébastien Coron and Ilya Kizhvatov. "An Efficient Method for Random Delay Generation in Embedded Software". In: *Cryptographic Hardware and Embedded Systems – CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Sept. 2009, pp. 156–170. ISBN: 978-3-642-04138-9. DOI: 10.1007/978-3-642-04138-9_12. URL: https://doi.org/10.1007/978-3-642-04138-9_12.

[38] Thomas M. Cover and Joy A. Thomas. "Entropy, Relative Entropy, and Mutual Information". In: *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, Sept. 2006, pp. 13–55. ISBN: 9780471748823. DOI: 10.1002/047174882X.ch2. URL: https://doi.org/10.1002/047174882X.ch2.

[39] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. "Leakage Assessment Through Neural Estimation of the Mutual Information". In: *Applied Cryptography and Network Security Workshops*. Ed. by Jianying Zhou, Mauro Conti, Chuadhry Mujeeb Ahmed, Man Ho Au, Lejla Batina, Zhou Li, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Weizhi Meng, Martín Ochoa, Stjepan Picek, Georgios Portokalidis, Cong Wang, and Kehuan Zhang. Lecture Notes in Computer Science. Rome, Italy: Springer International Publishing, 2020, pp. 144–162. ISBN: 978-3-030-61638-0. DOI: 10.1007/978-3-030-61638-0_9. URL: https://doi.org/10.1007/978-3-030-61638-0_9.

[40] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. "Revisiting Mutual Information Analysis: Multidimensionality, Neural Estimation and Optimality Proofs". In: *Journal of Cryptology* 36.4 (Aug. 2023), p. 38. ISSN: 1432-1378. DOI: 10.1007/s00145-023-09476-0. URL: https://doi.org/10.1007/s00145-023-09476-0.

[41] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 0932-4194, 1435-568X. DOI: 10.1007/bf02551274.

[42] Paweł Czyż, Frederic Grabowski, Julia E. Vogt, Niko Beerenwinkel, and Alexander Marx. "Beyond Normal: On the Evaluation of Mutual Information Estimators". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems* 36 (Dec. 2024), pp. 16957–16990. ISSN: 1049-5258. DOI: 10.5555/3666122.3666864. URL: https://dl.acm.org/doi/10.5555/3666122.3666864.

[43] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. 1st ed. Information Security and Cryptography. Copyright Information: Springer-Verlag Berlin Heidelberg 2002. Berlin, Heidelberg: Springer-Verlag, Feb. 2002, pp. XVII, 238. ISBN: 978-3-540-42580-9. DOI: 10.1007/978-3-662-04722-4. URL: https://doi.org/10.1007/978-3-662-04722-4.

[44] Daryl J. Daley and David Vere-Jones. "Scoring Probability Forecasts for Point Processes: The Entropy Score and Information Gain". In: *Journal of Applied Probability* 41.A (Dec. 2004), pp. 297–312. ISSN: 0021-9002. DOI: 10.1239/jap/1082552206.

[45] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. "On the Joint Security of Encryption and Signature in EMV". In: *Topics in Cryptology – CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer Berlin Heidelberg, 2012, pp. 116–135. ISBN: 978-3-642-27953-9. DOI: 10.1007/978-3-642-27954-6_8. URL: https://doi.org/10.1007/978-3-642-27954-6_8.

[46] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7.1 (2006), pp. 1–30. ISSN: 1532-4435.

[47] Luc Devroye, László Györfi, and Gábor Lugosi. "The Bayes Error". In: *A Probabilistic Theory of Pattern Recognition*. Vol. 31. New York, NY: Springer New York, 1996. Chap. 2, pp. 9–20. ISBN: 978-1-4612-0711-5. DOI: 10.1007/978-1-4612-0711-5_2. URL: https://doi.org/10.1007/978-1-4612-0711-5_2.

[48] Pedro Domingos and Michael Pazzani. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss". In: *Machine Learning* 29.2/3 (Nov. 1997), pp. 103–130. ISSN: 0885-6125. DOI: 10.1023/a:1007413511361.

[49] Jan Peter Drees, Pritha Gupta, Eyke Hüllermeier, Tibor Jager, Alexander Konze, Claudia Priesterjahn, Arunselvan Ramaswamy, and Juraj Somorovsky. "Automated Detection of Side Channels in Cryptographic Protocols". In: *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. Virtual Event, Republic of Korea: Association for Computing Machinery, Nov. 2021, pp. 169–180. ISBN: 9781450386579. DOI: 10.1145/3474369.3486868. URL: https://doi.org/10.1145/3474369.3486868.

[50] David Easter. "The impact of 'Tempest' on Anglo-American communications security and intelligence, 1943–1970". In: *Intelligence and National Security* 36.1 (Jan. 2021), pp. 1–16. DOI: 10.1080/02684527.2020.1798604. URL: https://doi.org/10.1080/02684527.2020.1798604.

[51] Emil Eirola, Amaury Lendasse, and Juha Karhunen. "Variable selection for regression problems using Gaussian mixture models to estimate mutual information". In: *2014 International Joint Conference on Neural Networks (IJCNN)*. Beijing, China: IEEE, July 2014, pp. 1606–1613. DOI: 10.1109/ijcnn.2014.6889561.

[52] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data". In: *arXiv preprint arXiv:2003.06505* (2020).

[53] Nick Erickson, Xingjian Shi, James Sharpnack, and Alexander Smola. "Multimodal AutoML for Image, Text and Tabular Data". In: *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 4786–4787. ISBN: 9781450393850. DOI: 10.1145/3534678.3542616.

[54] Sina Faezi, Rozhin Yasaei, Anomadarshi Barua, and Mohammad Abdullah Al Faruque. "Brain-Inspired Golden Chip Free Hardware Trojan Detection". In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 2697–2708. DOI: 10.1109/TIFS.2021.3062989. URL: https://ieeexplore.ieee.org/document/9366548.

[55] Stefan Falkner, Aaron Klein, and Frank Hutter. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: Proceedings of Machine Learning Research, July 2018, pp. 1437–1446. URL: http://proceedings.mlr.press/v80/falkner18a.html.

[56] Robert M Fano. *Transmission of Information: A Statistical Theory of Communications*. Cambridge, MA: The MIT Press, 1961.

[57] M. Feder and N. Merhav. "Relations between Entropy and Error Probability". In: *IEEE Transactions on Information Theory* 40.1 (Jan. 1994), pp. 259–266. ISSN: 0018-9448. DOI: 10.1109/18.272494. URL: https://doi.org/10.1109/18.272494.

[58] Dennis Felsch, Martin Grothe, Jörg Schwenk, Adam Czubak, and Marcin Szymanek. "The Dangers of Key Reuse: Practical Attacks on IPsec IKE". In: *27th USENIX Security Symposium (USENIX Security 18)*. Ed. by William Enck and Adrienne Porter Felt. Baltimore, MD, USA: USENIX Association, Aug. 2018, pp. 567–583. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/felsch.

[59] Andrey Feuerverger and Sheikh Rahman. "Some Aspects of Probability Forecasting". In: *Communications in Statistics - Theory and Methods* 21.6 (Jan. 1992), pp. 1615–1632. ISSN: 0361-0926. DOI: 10.1080/03610929208830868.

[60] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. "Auto-sklearn: Efficient and Robust Automated Machine Learning". In: *Automated Machine Learning: Meth-*

ods, Systems, Challenges. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Vol. 2520. Lecture Notes in Computer Science. Print ISBN: 978-3-030-05317-8. Cham: Springer International Publishing, 2019, pp. 113–134. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_6. URL: https://doi.org/10.1007/978-3-030-05318-5_6.

[61] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. "Initializing Bayesian Hyperparameter Optimization via Meta-Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas, USA: AAAI Press, 2015, pp. 1128–1135. ISBN: 0262511290. DOI: 10.1609/aaai.v29i1.9354. URL: https://doi.org/10.1609/aaai.v29i1.9354.

[62] Matthias Feurer, Jan N. Van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Müller, Joaquin Vanschoren, and Frank Hutter. "OpenML-Python: An Extensible Python API for OpenML". In: *Journal of Machine Learning Research* 22.1 (Jan. 2021). ISSN: 1532-4435.

[63] Telmo Silva Filho, Hao Song, Miquel Perello-Nieto, Raul Santos-Rodriguez, Meelis Kull, and Peter Flach. "Classifier Calibration: A Survey on How to Assess and Improve Predicted Class Probabilities". In: *Machine Learning* 112.9 (Sept. 2023), pp. 3211–3260. ISSN: 1573-0565. DOI: 10.1007/s10994-023-06336-7. URL: https://doi.org/10.1007/s10994-023-06336-7.

[64] R. A. Fisher. "On the Interpretation of χ 2 from Contingency Tables, and the Calculation of P". In: *Journal of the Royal Statistical Society* 85.1 (Jan. 1922), p. 87. ISSN: 0952-8385. DOI: 10.2307/2340521.

[65] Peter I. Frazier. "Optimization via Simulation with Bayesian Statistics and Dynamic Programming". In: *Proceedings of the 2012 Winter Simulation Conference (WSC)*. Ed. by Oliver Rose and Adelinde M. Uhrmacher. Winter Simulation Conference. Berlin, Germany: IEEE, Dec. 2012, pp. 1–16. DOI: 10.1109/WSC.2012.6465237. URL: https://doi.org/10.1109/WSC.2012.6465237.

[66] Luca Frittoli, Matteo Bocchi, Silvia Mella, Diego Carrera, Beatrice Rossi, Pasqualina Fragneto, Ruggero Susella, and Giacomo Boracchi. "Strengthening Sequential Side-Channel Attacks Through Change Detection". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.3 (June 2020), pp. 1–21. DOI: 10.13154/tches.v2020.i3.1-21. URL: https://doi.org/10.13154/tches.v2020.i3.1-21.

[67] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. "Affine Masking against Higher-Order Side Channel Analysis". In: *Selected Areas in Cryptography*. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Vol. 6544. Waterloo, Ontario, Canada: Springer Berlin Heidelberg, Aug. 2011, pp. 262–280. ISBN: 978-3-642-19574-7. DOI: 10.1007/978-3-642-19574-7_18.

[68] Dennis Funke. "Pushing the AutoSCA Tool to Picosecond Precision: Improving Timing Side Channel Detection". Bachelor's thesis. Sept. 2022. DOI: 10.13140/RG.2.2.33070.08005. URL: https://doi.org/10.13140/RG.2.2.33070.08005.

[69] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. "Efficient Estimation of Mutual Information for Strongly Dependent Variables". In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Guy Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, May 2015, pp. 277–286. URL: https://proceedings.mlr.press/v38/gao15.html.

[70] Catherine H. Gebotys, Simon Ho, and C. C. Tiu. "EM analysis of rijndael and ECC on a wireless java-based PDA". In: *Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems*. CHES'05. Edinburgh, UK: Springer-Verlag, 2005, pp. 250–264. ISBN: 3540284745. DOI: 10.1007/11545262_19. URL: https://doi.org/10.1007/11545262_19.

[71] Pieter Gijsbers, Marcos L. P. Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. "AMLB: an AutoML Benchmark". In: *Journal of Machine Learning Research* 25.101 (2024), pp. 1–65.

[72] Richard Gilmore, Neil Hanley, and Maire O'Neill. "Neural Network-Based Attack on a Masked Implementation of AES". In: *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 106–111. DOI: 10.1109/HST.2015.7140247. URL: https://doi.org/10.1109/HST.2015.7140247.

[73] Tilmann Gneiting and Adrian E. Raftery. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (Mar. 2007), pp. 359–378. ISSN: 0162-1459. DOI: 10.1198/016214506000001437.

[74] Aron Gohr and Sven Jacob and Werner Schindler. "CHES 2018 Side Channel Contest CTF - Solution of the AES Challenges". In: *IACR Cryptology ePrint Archive* (2019), p. 94. URL: https://eprint.iacr.org/2019/094.

[75] I. J. Good. "Rational Decisions". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. New York, NY: Springer New York, 1992, pp. 365–377. ISBN: 978-1-4612-0919-5. DOI: 10.1007/978-1-4612-0919-5_24.

[76] Google. *HTTPS Encryption on the Web*. Accessed: 2023-03-22. 2023. URL: https://transparencyreport.google.com/https/overview (visited on 03/22/2023).

[77] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning , (ICML)*. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1321–1330.

[78] Qian Guo, Vincent Grosso, and François-Xavier Standaert. "Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.3 (June 2020), pp. 1–21. DOI: 10.13154/tches.v2020.i3.1-21. URL: https://tches.iacr.org/index.php/TCHES/article/view/8581.

[79] Pritha Gupta, Jan Peter Drees, and Eyke Hüllermeier. "Automated Side-Channel Attacks using Black-Box Neural Architecture Search". In: *Proceedings of the 18th International Conference on Availability, Reliability and*

Security. ARES '23. Benevento, Italy: Association for Computing Machinery, 2023. ISBN: 9798400707728. DOI: 10.1145/3600160.3600161. URL: https://doi.org/10.1145/3600160.3600161.

[80] Pritha Gupta, Arunselvan Ramaswamy, Jan Drees, Eyke Hüllermeier, Claudia Priesterjahn, and Tibor 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. Virtual Event: SCITEPRESS - Science and Technology Publications, 2022, pp. 152–163. ISBN: 978-989-758-547-0. DOI: 10.5220/0010793000003116.

[81] Pritha Gupta, Marcel Wever, and Eyke Hüllermeier. "Information Leakage Detection through Approximate Bayes-optimal Prediction". In: *arXiv preprint arXiv:2401.14283* (2024).

[82] Jiaji He, Yiqiang Zhao, Xiaolong Guo, and Yier Jin. "Hardware Trojan Detection Through Chip-Free Electromagnetic Side-Channel Statistical Analysis". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25 (2017), pp. 2939–2948. DOI: 10.1109/TVLSI.2017.2727985. URL: https://ieeexplore.ieee.org/document/7994702.

[83] Malcolm Heath. *Scanning for CVE-2017-9841 Drops Precipitously*. 2024. URL: https://www.f5.com/labs/articles/threat-intelligence/sensor-intel-series-top-cves-july-2024 (visited on 11/07/2024).

[84] Malcolm Heath and David Warburton. *2024 DDoS Attack Trends*. 2024. URL: https://www.f5.com/labs/articles/threat-intelligence/2024-ddos-attack-trends (visited on 11/07/2024).

[85] M. Hellman and J. Raviv. "Probability of error, equivocation, and the Chernoff bound". In: *IEEE Transactions on Information Theory* 16.4 (July 1970), pp. 368–372. ISSN: 0018-9448. DOI: 10.1109/tit.1970.1054466.

[86] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. "Applications of Machine Learning Techniques in Side-Channel Attacks: A Survey". In: *Journal of Cryptographic Engineering* 10.2 (June 2020), pp. 135–162. ISSN: 2190-8516. DOI: 10.1007/s13389-019-00212-8. URL: https://doi.org/10.1007/s13389-019-00212-8.

[87] Benjamin Hettwer, Tobias Horn, Stefan Gehrer, and Tim Güneysu. "Encoding Power Traces as Images for Efficient Side-Channel Analysis". In: *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. San Jose, CA, USA: IEEE, 2020, pp. 46–56. DOI: 10.1109/HOST45689.2020.9300289. URL: https://doi.org/10.1109/HOST45689.2020.9300289.

[88] Annelie Heuser, Stjepan Picek, Sylvain Guilley, and Nele Mentens. "Lightweight Ciphers and Their Side-Channel Resilience". In: *IEEE Transactions on Computers* 69.10 (2020), pp. 1434–1448. DOI: 10.1109/TC.2017.2757921. URL: https://doi.org/10.1109/TC.2017.2757921.

[89] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. "TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=cp5PvcI6w8_.

[90] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. "Accurate predictions on small data with a tabular foundation model". In: *Nature* 637.8045 (Jan. 2025), pp. 319–326. ISSN: 1476-4687. DOI: 10.1038/s41586-024-08328-6. URL: https://doi.org/10.1038/s41586-024-08328-6.

[91] Sture Holm. "A Simple Sequentially Rejective Multiple Test Procedure". In: *Scandinavian Journal of Statistics* 6.2 (1979), pp. 65–70.

[92] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. "Machine learning in side-channel analysis: a first study". In: *Journal of Cryptographic Engineering* 1.4 (Dec. 2011), pp. 293–302. ISSN: 2190-8516. DOI: 10.1007/s13389-011-0023-x. URL: https://doi.org/10.1007/s13389-011-0023-x.

[93] Jeremy Howard and Sylvain Gugger. "Fastai: A Layered API for Deep Learning". In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020108.

[94] G. Hughes. "On the mean accuracy of statistical pattern recognizers". In: *IEEE Transactions on Information Theory* 14.1 (Jan. 1968), pp. 55–63. ISSN: 0018-9448. DOI: 10.1109/tit.1968.1054102.

[95] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. Lille, France: JMLR.org, 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[96] Tibor Jager, Sebastian Schinzel, and Juraj Somorovsky. "Bleichenbacher's Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption". In: *Computer Security – ESORICS 2012*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Vol. 7459. Lecture Notes in Computer Science. Pisa, Italy: Springer Berlin Heidelberg, 2012, pp. 752–769. ISBN: 978-3-642-33167-1. DOI: 10.1007/978-3-642-33167-1_43.

[97] Tibor Jager, Jörg Schwenk, and Juraj Somorovsky. "On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1185–1196. ISBN: 9781450338325. DOI: 10.1145/2810103.2813657. URL: https://doi.org/10.1145/2810103.2813657.

[98] David Jensen. "Data Snooping, Dredging and Fishing: The Dark Side of Data Mining - A SIGKDD99 Panel Report". In: *SIGKDD Explorations Newsletter* 1.2 (Jan. 2000), pp. 52–54. ISSN: 1931-0145. DOI: 10.1145/846183.846195. URL: https://doi.org/10.1145/846183.846195.

[99] Haifeng Jin. "Efficient Neural Architecture Search for Automated Deep Learning". AAI29241617. PhD thesis. Texas, USA: Texas A&M University, 2021. ISBN: 9798438744504. URL: https://oaktrust.library.tamu.edu/bitstream/handle/1969.1/193093/JIN-DISSERTATION-2021.pdf.

[100] Haifeng Jin, Qingquan Song, and Xia Hu. "Auto-Keras: An Efficient Neural Architecture Search System". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, Aug. 2019, pp. 1946–1956. ISBN: 9781450362016. DOI: 10.1145/3292500.3330648. URL: https://doi.org/10.1145/3292500.3330648.

[101] Kirthevasan Kandasamy, Akshay Krishnamurthy, Barnabýs Póczos, Larry Wasserman, and James M. Robins. "Nonparametric von Mises Estimators for Entropies, Divergences and Mutual Informations". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Montreal, Canada: Neural Information Processing Systems Foundation, Inc., 2015, pp. 397–405.

[102] Hubert Kario. "Everlasting ROBOT: The Marvin Attack". In: *Computer Security – ESORICS 2023: 28th European Symposium on Research in Computer Security, The Hague, The Netherlands, September 25–29, 2023, Proceedings, Part III*. Ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14137. Lecture Notes in Computer Science. The Hague, The Netherlands: Springer Nature Switzerland, 2024, pp. 243–262. ISBN: 978-3-031-51478-4. DOI: 10.1007/978-3-031-51479-1_13. URL: https://doi.org/10.1007/978-3-031-51479-1_13.

[103] Priyank Kashyap, Furkan Aydin, Seetal Potluri, Paul D. Franzon, and Aydin Aysu. "2Deep: Enhancing Side-Channel Attacks on Lattice-Based Key-Exchange via 2-D Deep Learning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40.6 (June 2021), pp. 1217–1229. ISSN: 0278-0070. DOI: 10.1109/TCAD.2020.3038701. URL: https://doi.org/10.1109/TCAD.2020.3038701.

[104] Evgnosia-Alexandra Kelesidis. "An Optimization of Bleichenbacher's Oracle Padding Attack". In: *Innovative Security Solutions for Information Technology and Communications, 14th International Conference, SecITC 2021*. Ed. by Peter Y. A. Ryan and Cristian Toma. Vol. 13195. Lecture Notes in Computer Science. Cham: Springer International Publishing, Nov. 2021, pp. 145–155. ISBN: 978-3-031-17510-7. DOI: 10.1007/978-3-031-17510-7_10. URL: https://doi.org/10.1007/978-3-031-17510-7_10.

[105] John Kelsey. "Compression and Information Leakage of Plaintext". In: *Fast Software Encryption*. Ed. by Joan Daemen and Vincent Rijmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 263–276. ISBN: 978-3-540-45661-2.

[106] Eike Kiltz, Adam O'Neill, and Adam Smith. "Instantiability of RSA-OAEP under Chosen-Plaintext Attack". In: *Journal of Cryptology* 30.3 (July 2017), pp. 889–919. ISSN: 0933-2790. DOI: 10.1007/s00145-016-9238-4. URL: https://doi.org/10.1007/s00145-016-9238-4.

[107] Vlastimil Klíma, Ondrej Pokorný, and Tomáš Rosa. "Attacking RSA-Based Sessions in SSL/TLS". In: *Cryptographic Hardware and Embedded Systems – CHES 2003*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Cologne, Germany: Springer, Heidelberg, Germany, 2003, pp. 426–440. ISBN: 978-3-540-45238-6. DOI: 10.1007/978-3-540-45238-6_33.

[108] Vlastimil Klíma and Tomáš Rosa. "Further Results and Considerations on Side Channel Attacks on RSA". In: *Cryptographic Hardware and Embedded Systems – CHES 2002*. Ed. by Burton S. Kaliski, Çetin Kaya Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 244–259. ISBN: 978-3-540-36400-9. DOI: 10.1007/3-540-36400-5_19. URL: https://doi.org/10.1007/3-540-36400-5_19.

[109] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology — CRYPTO '99, 19th Annual International Cryptology Conference*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 1999, pp. 388–397. ISBN: 978-3-540-48405-9. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1_25.

[110] Oluwasanmi Koyejo, Pradeep Ravikumar, Natara01 Nagarajan, and Inderjit S. Dhillon. "Consistent Multilabel Classification". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada: MIT Press, 2015, pp. 3321–3329. DOI: 10.5555/2969442.2969610.

[111] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physics Review E* 69 (6 June 2004), p. 066138. DOI: 10.1103/PhysRevE.69.066138. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.066138.

[112] Meelis Kull, Telmo Silva Filho, and Peter Flach. "Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Vol. 54. Proceedings of Machine Learning Research, Apr. 2017, pp. 623–631.

[113] Fabian Küppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. "Multivariate Confidence Calibration for Object Detection". In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Los Alamitos, CA, USA: IEEE, June 2020, pp. 1322–1330. DOI: 10.1109/cvprw50498.2020.00171.

[114] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. "A Machine Learning Approach Against a Masked AES — Reaching the Limit of Side-Channel Attacks with a Learning Model". In: *Journal of Cryptographic*

Engineering 5.2 (June 2015), pp. 123–139. ISSN: 2190-8516. DOI: 10.1007/s13389-014-0089-3. URL: https://doi.org/10.1007/s13389-014-0089-3.

[115] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. "Template Attacks versus Machine Learning Revisited and the Curse of Dimensionality in Side-Channel Analysis: Extended Version". In: *Journal of Cryptographic Engineering* 8.4 (Nov. 2018), pp. 301–313. ISSN: 2190-8516. DOI: 10.1007/s13389-017-0162-9. URL: https://doi.org/10.1007/s13389-017-0162-9.

[116] Liam Li, Kevin G. Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. "A System for Massively Parallel Hyperparameter Tuning". In: *Proceedings of Machine Learning and Systems 2020 (MLSys 2020)*. Ed. by Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze. Vol. 2. Austin, TX, USA: mlsys.org, Mar. 2020, pp. 230–246. URL: https://proceedings.mlsys.org/paper_files/paper/2020/file/a06f20b349c6cf09a6b171c71b88bbfc-Paper.pdf.

[117] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *Journal of Machine Learning Research* 18.185 (Jan. 2018), pp. 1–52. ISSN: 1532-4435. URL: http://jmlr.org/papers/v18/16-558.html.

[118] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. "How to Estimate the Success Rate of Higher-Order Side-Channel Attacks". In: *Cryptographic Hardware and Embedded Systems – CHES 2014*. Ed. by Lejla Batina and Matthew Robshaw. Vol. 8731. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 35–54. ISBN: 978-3-662-44709-3. DOI: 10.1007/978-3-662-44709-3_3. URL: https://doi.org/10.1007/978-3-662-44709-3_3.

[119] Lukas Maar, Stefan Gast, Martin Unterguggenberger, Mathias Oberhuber, and Stefan Mangard. "SLUBStick: Arbitrary Memory Writes through Practical Software Cross-Cache Attacks within the Linux Kernel". In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 4051–4068. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/maar-slubstick.

[120] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. "Breaking Cryptographic Implementations Using Deep Learning Techniques". In: *Security, Privacy, and Applied Cryptography Engineering*. Ed. by Claude Carlet, M. Anwar Hasan, and Vishal Saraswat. Cham: Springer International Publishing, 2016, pp. 3–26. DOI: 10.1007/978-3-319-49445-6_1.

[121] Felipe Maia Polo and Renato Vicente. "Effective sample size, dimensionality, and generalization in covariate shift adaptation". In: *Neural Computing and Applications* 35.25 (Jan. 2022), pp. 18187–18199. ISSN: 0941-0643, 1433-3058. DOI: 10.1007/s00521-021-06615-1.

[122] James Manger. "A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0". In: *Advances in Cryptology — CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2001, Proceedings*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 230–238. ISBN: 978-3-540-44647-7. DOI: 10.1007/3-540-44647-8_14. URL: https://doi.org/10.1007/3-540-44647-8_14.

[123] J. L. Massey. "Guessing and Entropy". In: *Proceedings of 1994 IEEE International Symposium on Information Theory*. 1994, p. 204. DOI: 10.1109/ISIT.1994.394764. URL: https://doi.org/10.1109/ISIT.1994.394764.

[124] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. "A Comprehensive Study of Deep Learning for Side-Channel Analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.1 (Nov. 2019), pp. 348–375. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i1.348-375. URL: https://doi.org/10.13154/tches.v2020.i1.348-375.

[125] R. J. McEliece and Z. Yu. "An Inequality on Entropy". In: *Proceedings of 1995 IEEE International Symposium on Information Theory*. 1995, p. 329. DOI: 10.1109/ISIT.1995.550316. URL: https://doi.org/10.1109/ISIT.1995.550316.

[126] Vitalik Melnikov, Eyke Hüllermeier, Daniel Kaimann, Bernd Frick, and Pritha Gupta. "Pairwise versus Pointwise Ranking: A Case Study". In: *Schedae Informaticae* (2016), pp. 73–83. DOI: 10.4467/20838476SI.16.006.6187.

[127] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)". In: *30th USENIX Security Symposium (USENIX Security 21)*. Baltimore, MD, USA: USENIX Association, Aug. 2021, pp. 213–230. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/merget.

[128] Robert Merget, Juraj Somorovsky, Nimrod Aviram, Craig Young, Janis Fliegenschmidt, Jörg Schwenk, and Yuval Shavitt. "Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities". In: *28th USENIX Security Symposium (USENIX Security 19)*. SEC'19. Santa Clara, CA, USA: USENIX Association, Aug. 2019, pp. 1029–1046. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/merget.

[129] Thomas S. Messerges. "Using Second-Order Power Analysis to Attack DPA Resistant Software". In: *Cryptographic Hardware and Embedded Systems — CHES 2000*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1965. Lecture

Notes in Computer Science. Worcester, Massachusetts, USA: Springer Berlin Heidelberg, Aug. 2000, pp. 238–251. ISBN: 978-3-540-44499-2. DOI: 10.1007/3-540-44499-8_19.

[130] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. "Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks". In: *23rd USENIX Security Symposium (USENIX Security 14)*. Ed. by Kevin Fu and Jaeyeon Jung. San Diego, CA, USA: USENIX Association, Aug. 2014, pp. 733–748. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-meyer.pdf.

[131] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. "Gyrophone: Recognizing Speech from Gyroscope Signals". In: *23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association, Aug. 2014, pp. 1053–1067. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky.

[132] Jan Mielniczuk and Joanna Tyrcha. "Consistency of multilayer perceptron regression estimators". In: *Neural Networks* 6.7 (Jan. 1993), pp. 1019–1022. ISSN: 0893-6080. DOI: 10.1016/s0893-6080(09)80011-7.

[133] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. "ML-Plan: Automated Machine Learning via Hierarchical Planning". In: *Machine Learning* 107.8-10 (Sept. 2018), pp. 1495–1515. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-018-5735-z. URL: https://doi.org/10.1007/s10994-018-5735-z.

[134] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. *This POODLE Bites: Exploiting The SSL 3.0 Fallback*. 2014. URL: https://www.openssl.org/~bodo/ssl-poodle.pdf.

[135] Kevin R. Moon, Kumar Sricharan, and Alfred O. Hero. "Ensemble Estimation of Generalized Mutual Information With Applications to Genomics". In: *IEEE Transactions on Information Theory* 67.9 (2021), pp. 5963–5996. DOI: 10.1109/TIT.2021.3100108.

[136] Thorben Moos, Felix Wegener, and Amir Moradi. "DL-LA: Deep Learning Leakage Assessment: A Modern Roadmap for SCA Evaluations". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 552–598. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i3.552-598. URL: https://doi.org/10.46586/tches.v2021.i3.552-598.

[137] Maria Mushtaq, Ayaz Akram, Muhammad Khurram Bhatti, Maham Chaudhry, Vianney Lapotre, and Guy Gogniat. "NIGHTs-WATCH: A Cache-Based Side-Channel Intrusion Detector Using Hardware Performance Counters". In: *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*. Los Angeles, California: Association for Computing Machinery, June 2018. ISBN: 9781450365000. DOI: 10.1145/3214292.3214293.

[138] Claude Nadeau. "Inference for the Generalization Error". In: *Machine Learning* 52.3 (Sept. 2003), pp. 239–281. ISSN: 0885-6125. DOI: 10.1023/a:1024068626366.

[139] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. Denver, Colorado, USA: Association for Computing Machinery, July 2016, pp. 485–492. ISBN: 9781450342063. DOI: 10.1145/2908812.2908918.

[140] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 2015). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v29i1.9602.

[141] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, 06jie Bai, and Soumith Chintala. "PyTorch: An Imperative Style,

High-Performance Deep Learning Library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[142] Thomas Perianin, Sebastien Carré, Victor Dyseryn, Adrien Facon, and Sylvain Guilley. "End-to-end automated cache-timing attack driven by machine learning". In: *Journal of Cryptographic Engineering* 11.2 (June 2020), pp. 135–146. ISSN: 2190-8508, 2190-8516. DOI: 10.1007/s13389-020-00228-5.

[143] Guilherme Perin, Łukasz Chmielewski, and Stjepan Picek. "Strength in Numbers: Improving Generalization with Ensembles in Machine Learning-Based Profiled SCA". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.4 (Aug. 2020), pp. 337–364. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i4.337-364. URL: https://doi.org/10.13154/tches.v2020.i4.337-364.

[144] Guilherme Perin, Lichao Wu, and Stjepan Picek. *AISY - Deep Learning-based Framework for Side-channel Analysis*. Cryptology ePrint Archive, Report 2021/357. https://eprint.iacr.org/2021/357 and https://github.com/AISyLab/AISY_Framework. 2021. URL: https://eprint.iacr.org/2021/357.

[145] Karlson Pfannschmidt, Pritha Gupta, Björn Haddenhorst, and Eyke Hüllermeier. "Learning context-dependent choice functions". In: *International Journal of Approximate Reasoning* 140 (Jan. 2022), pp. 116–155. ISSN: 0888-613X. URL: https://www.sciencedirect.com/science/article/pii/S0888613X21001614.

[146] Karlson Pfannschmidt, Pritha Gupta, and Eyke Hüllermeier. *Deep Architectures for Learning Context-dependent Ranking Functions*. 2018. arXiv: 1803.05796 [stat.ML]. URL: https://arxiv.org/abs/1803.05796.

[147] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. "Template Attack versus Bayes Classifier". In: *Journal of Cryptographic Engineering* 7.4 (Nov. 2017), pp. 343–351. ISSN: 2190-8516. DOI: 10.1007/s13389-017-0172-7. URL: https://doi.org/10.1007/s13389-017-0172-7.

[148] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. "The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.1 (Nov. 2018), pp. 209–237. ISSN: 2569-2925. DOI: 10.13154/tches.v2019.i1.209-237. URL: https://doi.org/10.13154/tches.v2019.i1.209-237.

[149] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A. Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. "Side-Channel Analysis and Machine Learning: A Practical Perspective". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 4095–4102. DOI: 10.1109/IJCNN.2017.7966373. URL: https://doi.org/10.1109/IJCNN.2017.7966373.

[150] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. "SoK: Deep Learning-Based Physical Side-Channel Analysis". In: *ACM Computing Surveys* 55.11 (Feb. 2023). ISSN: 0360-0300. DOI: 10.1145/3569577. URL: https://doi.org/10.1145/3569577.

[151] John C. Platt. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In: *Advances in large margin classifiers* 10.3 (1999), pp. 61–74.

[152] John C. Platt. "Probabilities for SV Machines". In: *Advances in Large-Margin Classifiers*. Cambridge: MIT Press, 2000, pp. 61–73. ISBN: 9780262283977. DOI: 10.7551/mitpress/1113.001.0001.

[153] Felipe Maia Polo and Felipe Leno Da Silva. *InfoSelect - Mutual Information Based Feature Selection in Python*. 2020.

[154] David Martin Powers. "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.

[155] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. "Unifying Leakage Models on a Rényi Day". In: *Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference, Santa*

Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I. Ed. by Alexandra Boldyreva and Daniele Micciancio. Santa Barbara, CA, USA: Springer International Publishing, 2019, pp. 683–712. ISBN: 978-3-030-26948-7. DOI: 10.1007/978-3-030-26948-7_24. URL: https://doi.org/10.1007/978-3-030-26948-7_24.

[156] Claudia Priesterjahn, Jan Peter Drees, Pritha Gupta, and Simon Oberthur. *Anwendung von maschinellem Lernen zum automatischen Erkennen von Padding-Orakel-Seitenkanälen*. de. Text/Conference Paper. 2023. URL: https://dl.gi.de/handle/20.500.12116/43481,.

[157] Emmanuel Prouff. "DPA Attacks and S-Boxes". In: *Fast Software Encryption*. Ed. by Henri Gilbert and Helena Handschuh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 424–441. ISBN: 978-3-540-31669-5.

[158] Zhenyue Qin and Dongwoo Kim. "Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator". In: *CoRR* abs/1911.10688 (2019). arXiv: 1911.10688.

[159] Mick G.D. Remmerswaal, Lichao Wu, Sébastien Tiran, and Nele Mentens. "AutoPOI: Automated Points of Interest Selection for Side-Channel Analysis". In: *Journal of Cryptographic Engineering* 14.3 (Sept. 2024), pp. 463–474. DOI: 10.1007/s13389-023-00328-y. URL: https://doi.org/10.1007/s13389-023-00328-y.

[160] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. "A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions". In: *ACM Computing Surveys (CSUR)* 54.4 (May 2021), article 76, 1–34. ISSN: 0360-0300. DOI: 10.1145/3447582. URL: https://doi.org/10.1145/3447582.

[161] Alfréd Rényi. "On measures of entropy and information". In: *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*. Vol. 4. University of California Press. 1961, pp. 547–562.

[162] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. RFC 2246 (Historic). RFC. Obsoleted by RFC 4346, updated by multiple RFCs. Fremont, CA, USA, Jan. 1999. DOI: 10.17487/RFC2246. URL: https://www.rfc-editor.org/rfc/rfc2246.txt.

[163] B. Kaliski. *PKCS #1: RSA Encryption Version 1.5*. RFC 2313 (Informational). RFC. Obsoleted by RFC 2437. Fremont, CA, USA, Mar. 1998. DOI: 10.17487/RFC2313. URL: https://www.rfc-editor.org/rfc/rfc2313.txt.

[164] B. Kaliski and J. Staddon. *PKCS #1: RSA Cryptography Specifications Version 2.0*. RFC 2437 (Informational). RFC. Obsoleted by RFC 3447. Fremont, CA, USA, Oct. 1998. DOI: 10.17487/RFC2437. URL: https://www.rfc-editor.org/rfc/rfc2437.txt.

[165] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*. RFC 4346 (Historic). RFC. Obsoleted by RFC 5246, updated by multiple RFCs. Fremont, CA, USA, Apr. 2006. DOI: 10.17487/RFC4346. URL: https://www.rfc-editor.org/rfc/rfc4346.txt.

[166] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). RFC. Obsoleted by RFC 8446, updated by multiple RFCs. Fremont, CA, USA, Aug. 2008. DOI: 10.17487/RFC5246. URL: https://www.rfc-editor.org/rfc/rfc5246.txt.

[167] A. Freier, P. Karlton, and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. RFC 6101 (Historic). RFC. Fremont, CA, USA, Aug. 2011. DOI: 10.17487/RFC6101. URL: https://www.rfc-editor.org/rfc/rfc6101.txt.

[168] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446 (Proposed Standard). RFC. Fremont, CA, USA, Aug. 2018. DOI: 10.17487/RFC8446. URL: https://www.rfc-editor.org/rfc/rfc8446.txt.

[169] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. "Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 677–707. ISSN: 2569-2925. DOI: 10.46586/tches.v2021.i3.677-707. URL: https://tches.iacr.org/index.php/TCHES/article/view/8989.

[170] Olivier Rioul. "This is IT: A Primer on Shannon's Entropy and Information". In: *Information Theory: Poincaré Seminar 2018*. Ed. by Bertrand Duplantier and Vincent Rivasseau. Cham: Springer International Publishing, 2021, pp. 49–86. ISBN: 978-3-030-81480-9. DOI: 10.1007/978-3-030-81480-9_2.

[171] Olivier Rioul. "Variations on a Theme by Massey". In: *IEEE Transactions on Information Theory* 68.5 (2022), pp. 2813–2828. DOI: 10.1109/TIT.2022.3141264. URL: https://doi.org/10.1109/TIT.2022.3141264.

[172] Eyal Ronen, Robert Gillham, Daniel Genkin, Adi Shamir, David Wong, and Yuval Yarom. "The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations". In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA: IEEE Computer Society Press, 2019, pp. 435–452. DOI: 10.1109/SP.2019.00062. URL: https://doi.org/10.1109/SP.2019.00062.

[173] Mark S. Roulston and Leonard A. Smith. "Evaluating Probabilistic Forecasts Using Information Theory". English. In: *Monthly Weather Review* 130.6 (June 2002), pp. 1653–1660.

[174] Joeri de Ruiter and Erik Poll. "Protocol State Fuzzing of TLS Implementations". In: *24th USENIX Security Symposium (USENIX Security 15)*. Proceedings of the 24th USENIX Conference on Security Symposium. SEC'15. Washington, D.C., USA: USENIX Association, Aug. 2015, pp. 193–206. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter.

[175] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones." In: *NDSS*. Vol. 11. 2011, pp. 17–33.

[176] David Schubert, Pritha Gupta, and Marcel Wever. "Meta-learning for Automated Selection of Anomaly Detectors for Semi-supervised Datasets". In: *Advances in Intelligent Data Analysis XXI*. Ed. by Bruno Crémilleux, Sibylle Hess, and Siegfried Nijssen. Cham: Springer Nature Switzerland, 2023, pp. 392–405. ISBN: 978-3-031-30047-9.

[177] Matthias Schulz, Patrick Klapper, Matthias Hollick, Erik Tews, and Stefan Katzenbeisser. "Trust The Wire, They Always Told Me! On Practical Non-Destructive Wire-Tap Attacks Against Ethernet". In: *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '16. Darmstadt, Germany: Association for Computing Machinery, 2016, pp. 43–48. ISBN: 9781450342704. DOI: 10.1145/2939918.2940650. URL: https://doi.org/10.1145/2939918.2940650.

[178] Asaf Shabtai, Yuval Elovici, and Lior Rokach. *A Survey of Data Leakage Detection and Prevention Solutions*. 1st ed. SpringerBriefs in Computer Science. New York, NY, USA: Springer US, Mar. 2012, pp. VIII, 92. ISBN: 978-1-4614-2052-1. DOI: 10.1007/978-1-4614-2053-8. URL: https://doi.org/10.1007/978-1-4614-2053-8.

[179] Aria Shahverdi, Mahammad Shirinov, and Dana Dachman-Soled. "Database Reconstruction from Noisy Volumes: A Cache Side-Channel Attack on SQLite". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 1019–1035. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/shahverdi.

[180] Mahdi Jafari Siavoshani, Amirhossein Khajehpour, Amirmohammad Ziaei Bideh, Amirali Gatmiri, and Ali Taheri. "Machine learning interpretability meets TLS fingerprinting". In: *Soft Computing* 27.11 (June 2023), pp. 7191–7208. ISSN: 1433-7479. DOI: 10.1007/s00500-023-07949-9. URL: https://doi.org/10.1007/s00500-023-07949-9.

[181] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations (ICLR 2015)*. Conference Track Proceedings. San Diego, CA, USA: Computational and Biological Learning Society, 2015, pp. 1–14. DOI: 10.48550/arXiv.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[182] Shashank Singh and Barnabás Póczos. "Finite-Sample Analysis of Fixed-k Nearest Neighbor Density Functional Estimators". In: *Advances in Neural Information Processing Systems 29*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. Barcelona, Spain: Neural Information Processing Systems Foundation, Inc., 2016, pp. 1217–1225.

[183] Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. "Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices". In: *IEEE Communications Surveys & Tutorials* 20.1 (2018), pp. 465–488. DOI: 10.1109/COMST.2017.2779824.

[184] François-Xavier Standaert, Tal Malkin, and Moti Yung. "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks". In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Cologne, Germany: Springer Berlin Heidelberg, Apr. 2009, pp. 443–461. ISBN: 978-3-642-01001-9. DOI: 10.1007/978-3-642-01001-9_26. URL: https://doi.org/10.1007/978-3-642-01001-9_26.

[185] D. Tebbe and S. Dwyer. "Uncertainty and the probability of error (Corresp.)" In: *IEEE Transactions on Information Theory* 14.3 (May 1968), pp. 516–518. ISSN: 0018-9448. DOI: 10.1109/tit.1968.1054135.

[186] Aika Terada, Koji Tsuda, and Jun Sese. "Fast Westfall-Young permutation procedure for combinatorial regulation discovery". In: *2013 IEEE International Conference on Bioinformatics and Biomedicine*. 2013, pp. 153–158. DOI: 10.1109/BIBM.2013.6732479.

[187] Sergios Theodoridis and Konstantinos Koutroumbas. "Chapter 5 - Feature Selection". In: *Pattern Recognition (Fourth Edition)*. Ed. by Sergios Theodoridis and Konstantinos Koutroumbas. 4th. Boston: Academic Press, 2009, pp. 261–322. ISBN: 978-1-59749-272-0. DOI: 10.1016/B978-1-59749-272-0.50007-4. URL: https://www.sciencedirect.com/science/article/pii/B9781597492720500074.

[188] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. "Success through Confidence: Evaluating the Effectiveness of a Side-Channel Attack". In: *Cryptographic Hardware and Embedded Systems - CHES 2013*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 7966. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 21–36. ISBN: 978-3-642-40349-1. DOI: 10.1007/978-3-642-40349-1_2.

[189] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 847–855. ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2487629. URL: https://doi.org/10.1145/2487575.2487629.

[190] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. "OpenML: Networked Science in Machine Learning". In: *SIGKDD explorations newsletter* 15.2 (June 2014), pp. 49–60. ISSN: 1931-0145. DOI: 10.1145/2641190.2641198.

[191] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 831–838. ISBN: 1558602224.

[192] Serge Vaudenay. "Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS..." In: *Advances in Cryptology — EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 534–545. ISBN: 978-3-540-46035-0. DOI: 10.1007/3-540-46035-7_35.

[193] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2.

[194] A.D. Walters and E. Kedaigle. "SLEAK: A Side-Channel Leakage Evaluator and Analysis Kit". In: *Technical Paper* (Nov. 2014). Accessed on 24 September 2020. URL: https://www.mitre.org/publications/technical-papers/sleak-a-side-channel-leakage-evaluator-and-analysis-kit.

[195] Andreas Walz and Axel Sikora. "Exploiting Dissent: Towards Fuzzing-Based Differential Black-Box Testing of TLS Implementations". In: *IEEE Transactions on Dependable and Secure Computing* 17.2 (2020), pp. 278–291. DOI: 10.1109/TDSC.2017.2763947.

[196] David Warburton and Sander Vinberg. *The 2021 TLS Telemetry Report*. 2021. URL: https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report (visited on 11/07/2024).

[197] A. F. Webster and S. E. Tavares. "On the Design of S-Boxes". In: *Advances in Cryptology — CRYPTO '85 Proceedings*. Ed. by Hugh C. Williams. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 523–534. ISBN: 978-3-540-39799-1. DOI: 10.1007/3-540-39799-X_41.

[198] Carolyn Whitnall and Elisabeth Oswald. "A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 316–334. ISBN: 978-3-642-22792-9.

[199] Manfred von Willich. "A Technique with an Information-Theoretic Basis for Protecting Secret Data from Differential Power Attacks". In: *Cryptography and Coding: Proceedings of the 8th IMA International Conference*. Ed. by Bahram Honary. Vol. 2260. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, Dec. 2001, pp. 44–62. ISBN: 978-3-540-45325-3. DOI: 10.1007/3-540-45325-3_6.

[200] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. "Revisiting a Methodology for Efficient article Architectures in Profiling Attacks". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.3 (June 2020), pp. 147–168. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i3.147-168. URL: https://doi.org/10.13154/tches.v2020.i3.147-168.

[201] Lichao Wu, Guilherme Perin, and Stjepan Picek. "I Choose You: Automated Hyperparameter Tuning for Deep Learning-Based Side-Channel Analysis". In: *IEEE Transactions on Emerging Topics in Computing* 12.2 (2024), pp. 546–557. ISSN: 2168-6750. DOI: 10.1109/TETC.2022.3218372. URL: https://www.openaccess.nl/en/you-share-we-take-care.

[202] M. Wu, S. McCamant, P.-C. Yew, and A. Zhai. "PREDATOR: A Cache Side-Channel Attack Detector Based on Precise Event Monitoring". In: *2022 IEEE International Symposium on Secure and Private Execution*

Environment Design (SEED). 2022 IEEE International Symposium on Secure and Private Execution Environment Design (SEED). 2022, pp. 25–36. DOI: 10.1109/SEED55351.2022.00010.

[203] Yuan Xiao, Mengyuan Li, Sanchuan Chen, and Yinqian Zhang. "STACCO: Differentially Analyzing Side-Channel Traces for Detecting SSL/TLS Vulnerabilities in Secure Enclaves". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 859–874. ISBN: 9781450349468. DOI: 10.1145/3133956.3134016. URL: https://doi.org/10.1145/3133956.3134016.

[204] Bianca Zadrozny and Charles Elkan. "Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers". In: *Proceedings of the 18th International Conference on Machine Learning , (ICML)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 609–616. ISBN: 1558607781.

[205] Bianca Zadrozny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada: Association for Computing Machinery, 2002, pp. 694–699. ISBN: 158113567X. DOI: 10.1145/775047.775151.

[206] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. "Methodology for Efficient article Architectures in Profiling Attacks". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.1 (Nov. 2019), pp. 1–36. ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i1.1-36. URL: https://doi.org/10.13154/tches.v2020.i1.1-36.

[207] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. "A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.3 (June 2020), pp. 73–96. ISSN: 2569-2925. DOI: 10.46586/tches.v2020.i3.73-96.

[208] Jie Zhang, Xiaolong Zheng, Zhanyong Tang, Tianzhang Xing, Xiaojiang Chen, Dingyi Fang, Rong Li, Xiaoqing Gong, and Feng Chen. "Privacy Leakage in Mobile Sensing: Your Unlock Passwords Can Be Leaked through Wireless Hotspot Functionality". In: *Mobile Information Systems* 2016.1 (Jan. 2016), p. 8793025. ISSN: 1574-017X. DOI: 10.1155/2016/8793025. URL: https://doi.org/10.1155/2016/8793025.

[209] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. "Cross-Tenant Side-Channel Attacks in PaaS Clouds". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. CCS '14. Scottsdale, AZ, USA: Association for Computing Machinery, 2014, pp. 990–1003. ISBN: 9781450329576. DOI: 10.1145/2660267.2660356. URL: https://dl.acm.org/doi/10.1145/2660267.2660356.

[210] Ming-Jie Zhao, Narayanan Edakunni, Adam Pocock, and Gavin Brown. "Beyond Fano's Inequality: Bounds on the Optimal F-Score, BER, and Cost-Sensitive Risk and Their Implications". In: *Journal of Machine Learning Research* 14.32 (2013), pp. 1033–1090.

[211] Yongbin Zhou and Dengguo Feng. "Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing". In: *IACR Cryptology ePrint Archive* (Oct. 2005). URL: http://eprint.iacr.org/2005/388.

[212] Marc-André Zöller and Marco F. Huber. "Benchmark and Survey of Automated Machine Learning Frameworks". In: *Journal of Artificial Intelligence Research* 70 (May 2021), pp. 409–472. ISSN: 1076-9757. DOI: 10.1613/jair.1.11854. URL: https://www.jair.org/index.php/jair/article/view/11854.

A. Appendix

The appendix provides detailed results and comprehensive implementation details for the experiments conducted.

A.1. Implementation Details

This section presents the implementation details, including parameters for performing HPO and the Python packages used for running the experiments conducted to detect ILDs in OpenSSL TLS Servers and AES-encrypted Systems, as described in Chapter 5 and Chapter 6, respectively. Additionally, the details of the experiments conducted to compare the MI estimation methods described in Chapter 4 are outlined as well.

Hyperparameter Optimization The models, learning algorithms, and their respective hyperparameters and range values used for HPO in the AutoGluon tool are outlined in Table A.1. The 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. The search space in AutoGluon includes tree-based ensemble models such as RF, XT, and GBM algorithms (LightGBM, CatBoost, and XGBoost), as well as MLP implemented using `PyTorch` and the trained MLP provided by `fastai` [132, 93, 18]. For high-dimensional datasets ($d > 100$), dimensionality reduction techniques are

Table A.1.: Hyperparameter ranges for AutoML tools: AutoGluon models and TabPFN including the MI estimation baseline approaches (Gaussian mixture model (GMM), MINE, and PC-SOFTMAX)

AutoGluon								
Tree-based Ensemble Models								
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
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)								
	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								
Learner	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

applied to TabPFN and GMM. The detailed hyperparameters and range values for TabPFN, AutoGluon models, and the baseline MI estimators, including GMM, MINE, and PC-Softmax, are provided in Table A.1.

A.1.1. MI Estimation Approaches

The synthetic datasets using the MVN distribution, including the ground-truth MI calculation, are generated using the `multivariate_normal` and `ortho_group` functions provided by `scipy` [193]. The HPO process is performed using Bayesian optimization (BO) as implemented by the `BayesSearchCV` in `scikit-optimize`[1]. The code for implementing all MI estimation methods and synthetic dataset generation processes is publicly available on GitHub[2], along with the scripts used for experiments[3].

The MI estimation methods leverage TabPFN and AutoGluon, as detailed in Section 2.2.4, to induce the Bayes predictor. Additionally, the implementations of five calibration techniques for the Cal Log-Loss approaches are provided by `netcal` [113]. The `InfoSelect` library is extended to support multiple covariance structures (diagonal, spherical, and tied)[1], with dimensionality reduction applied for high-dimensional datasets ($d > 100$) to implement the GMM approach. The architectures and objective function for the MINE approach are implemented using `PyTorch` [141], while the MLP architectures, including the categorical cross-entropy (CCE) loss for the PC-Softmax approach, are implemented using `Keras` [33].

[1] `https://github.com/scikit-optimize/scikit-optimize`
[2] `https://github.com/LeakDetectAI/AutoMLQuantILDetect`
[3] `https://github.com/LeakDetectAI/automl-qild-experiments`

A.1.2. Automating ILD in OpenSSL TLS Servers

The MI-based ILD approaches employ the OTT statistical test on estimated MIs, as discussed in Section 3.2.2. The classification-based ILD approaches, such as PTT-Majority, FET-Mean, and FET-Median, induce the Bayes predictor using TabPFN and AutoGluon, as detailed in Section 2.2.4. The statistical tests, including FET, PTT, OTT, and Holm-Bonferroni correction, are implemented using `scipy` [193]. The 10 IL-Datasets corresponding to each time delay and error message are converted into a multi-class classification dataset containing 11 padding classes, including correctly padded messages (labeled as 0).These datasets, designed by [190], are uploaded to OpenML[4] using `openml` [62]. The documented code for implementing all ILD approaches and parsing OpenML real IL-Datasets is available on GitHub[2], with the scripts for experiments hosted on GitHub[3].

A.1.3. Automating ILD in AES-encrypted Systems

To implement NAS for different search strategies, search spaces, and input shapes, the `AutoModel`, `DenseBlock`, `ConvBlock`, and `ClassificationHead` classes from the `AutoKeras` library [100] are extended. The code for experiments and plot generation is publicly available on GitHub[5].

Baseline Architectures

Fixed baseline architectures are trained for comparison with the NAS approach. The ASCAD architecture proposed by Benadjila et al. (2020) [14] and the ZAID architectures by Zaid et al. (2019) [206] are utilized. The ASCAD baseline is a CNN model inspired by the VGG-16 CNN model [181], as shown in Figure 6.1a.

[4] `https://www.openml.org/s/417`
[5] `https://github.com/LeakDetectAI/deep-learning-sca`

For the ASCAD_r datasets, the baselines for corresponding fixed-key versions are used. For the CHES CTF dataset, the deepest CNN proposed by Zaid et al. (2019) [206] is employed, as shown in Figure 6.1b.

Computing Hardware and Runtime

The experiments necessitate the training of millions of CNN models, requiring thousands of hours of GPU time. These experiments are executed in parallel on a supercomputer equipped with GPU nodes, enabling the completion of the entire parameter study within a few weeks. The GPU nodes consist of two AMD Milan 7763 CPUs operating at 2.45 GHz, 512 GB of main memory, and four Nvidia A100 GPUs connected via NVLink and equipped with 40 GB HBM2 GPU memory. A single NAS experiment, which identifies the best architecture through repeated intermediate model training followed by a final model training, requires less than 2 days on these shared GPU nodes. All search strategies are allocated a budget of testing 1000 architectures. The RANDOM and BAYESIAN strategies consistently utilize the entire budget, whereas the GREEDY strategy uses only a portion of the budget, resulting in a maximum runtime of 5 hours. The preemption technique employed by HYPERBAND reduces the maximum runtime to 12 hours. Some experiments are also conducted on consumer hardware, represented by a $2500 gaming PC, where the most extended experiment takes 22 hours. This gaming PC is equipped with an AMD Ryzen 9 3950X CPU operating at 3.5 GHz, 64 GB of main memory, and a single GeForce RTX 2080 Super GPU featuring 8 GB GDDR6 GPU memory. The runtime for an example experiment on the AES_HD dataset on this gaming PC is 22 hours for RANDOM, 3 hours and 45 minutes for HYPERBAND, 2 hours and 55 minutes for GREEDY, and 10 hours and 12 minutes for BAYESIAN. This runtime is a reasonable estimate for an attacker's lower bound of computational resources to allocate to a single attack.However, attackers with better-equipped systems are likely to exceed these constraints significantly.

A.2. ILD Performance on Error Code Datasets

This section evaluates the performance of the ILD approaches against state-of-the-art methods. Heatmaps with a fixed rejection threshold of 5 ($\tau = 5$) are used to illustrate key performance metrics, including Accuracy, FPR, and FNR.These metrics provide a detailed comparison of ILD approaches on the 9 OpenSSL TLS server datasets, as shown in Figure A.1.

A.2.1. Non-vulnerable OpenSSL TLS Servers

The performance of ILD approaches is focused on three non-vulnerable OpenSSL TLS servers—OpenSSL-1.1.1k, OpenSSL-0.9.7b, and OpenSSL-1.1.1t—while considering class imbalance cases ($r \in 0.1, 0.3, 0.5$). These servers are represented as configurations less susceptible to Bleichenbacher SCAs, allowing detection trends to be observed in scenarios where actual ILs are minimal or non-existent.

OpenSSL-1.1.1k

Out of MI-based ILD approaches using Log-Loss or Cal Log-Loss and classification-based techniques including FET-Mean, FET-Median and PTT-Majority maintain high accuracy (100 % at $r = 0.5$) with low FPR, demonstrating robustness even in imbalanced datasets, with AutoGluon outperforming TabPFN. However, approaches such as PTT-Random and Mid-Point exhibit high FPR (up to 90 %), leading to false positives for imbalanced datasets. TabPFN FET-Mean and PTT-Majority reach 100 % accuracy consistently, while baseline methods, such as GMM, fall below 10 %, reflecting random-like performance in the absence of true ILs. While, the MINE and probabilitiy-corrected softmax (PC-softmax) detects over 90 % of the ILs.

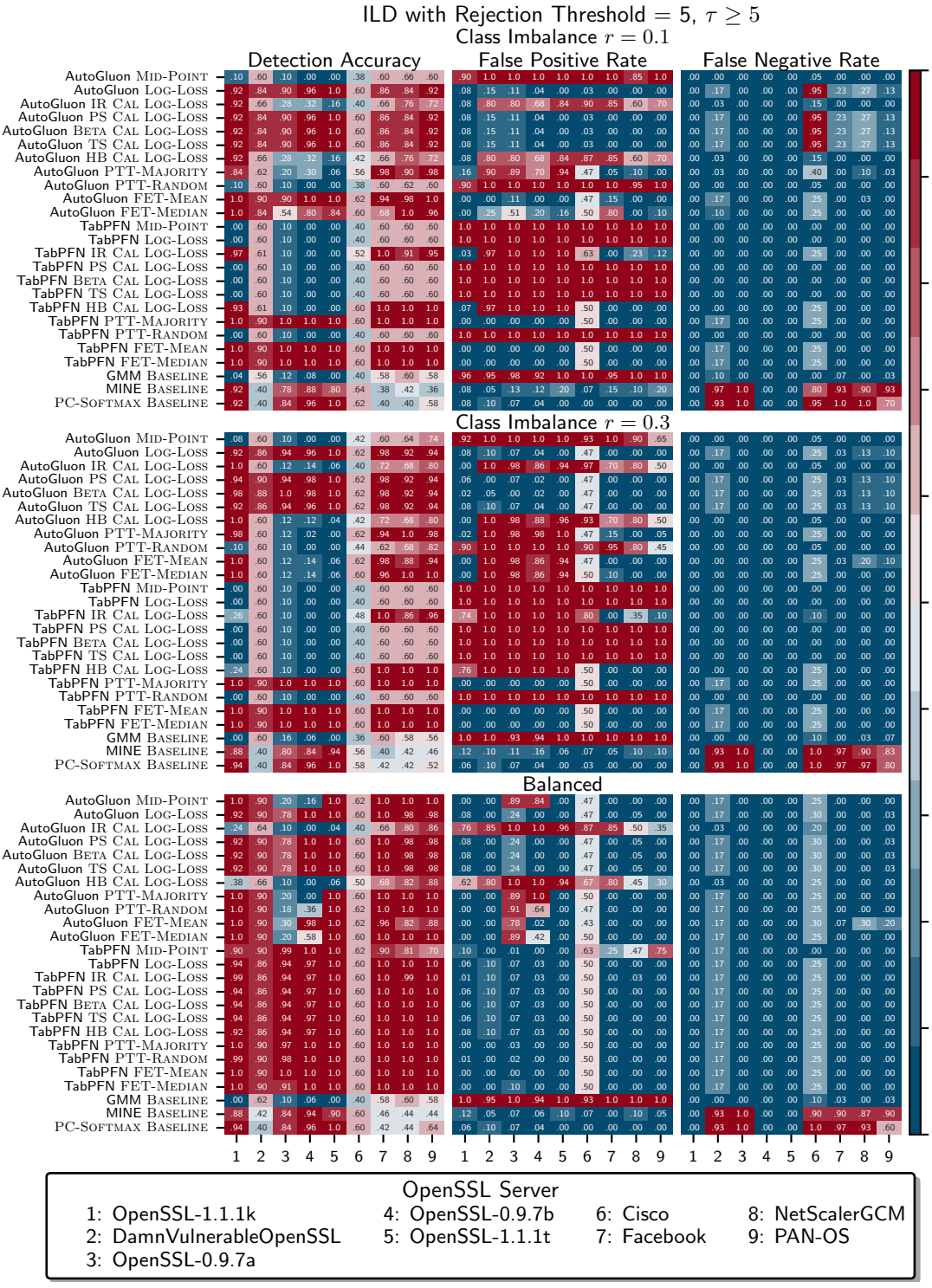


Figure A.1.: Performance of ILD approaches for all OpenSSL TLS Servers

OpenSSL-0.9.7b

OpenSSL-0.9.7b similarly highlights the efficacy of Cal Log-Loss, with TabPFN and AutoGluon calibration-based methods maintaining near-perfect accuracy at $r = 0.5$, achieving low FPR and consistent detection rates for balanced datasets. For imbalanced cases ($r = 0.1$ and 0.3), FET approaches under AutoGluon still perform well, while PTT-Random and Mid-Point continue to struggle, with FPR exceeding 90 %. Baseline methods like GMM show significantly lower performance with below 10 % detection accuracy, with MINE and PC-softmax detecting over 90 % of the ILs.

OpenSSL-1.1.1t

For OpenSSL-1.1.1t, AutoGluon Log-Loss (calibrated Cal Log-Loss), TabPFN FET-based and TabPFN PTT-Majority approaches almost all achieve 100 % accuracy across all class imbalances, with minimal false positives, reflecting consistent calibration effectiveness. Here, AutoGluon Log-Loss and TabPFN FET-Mean outperform other approaches, including MINE and PC-softmax baseline methods, which yield detection rates of more than 90 % in non-vulnerable environments. Notably, Mid-Point and PTT-Random report high FPR for systems generating imbalanced datasets, confirming their susceptibility to false positives in scenarios without substantial ILs.

A.2.2. Vulnerable OpenSSL TLS Servers

In this subsection, the performance of ILD techniques is analyzed on two highly vulnerable OpenSSL TLS servers—OpenSSL-0.9.7a and DamnVulnerableOpenSSL—across various class imbalances ($r \in 0.1, 0.3, 0.5$). Additionally, different imitation servers—Cisco, Facebook, NetScalerGCM, and PAN-OS—are analyzed to identify numerous ILs revealed via real-world side channels across public web servers, particularly in closed-source TLS server implementations.

These servers present significant ILs that render them susceptible to Bleichenbacher SCAs, which allows the detection effectiveness of various methods to be examined in environments with prevalent vulnerabilities.

OpenSSL-0.9.7a

For OpenSSL-0.9.7a, the MI-based AutoGluon Cal Log-Loss and TabPFN classification-based (FET-Mean, FET-Median, PTT-Majority) ILD approaches demonstrate strong detection rates, especially in balanced datasets ($r = 0.5$), achieving near-perfect accuracy (100 %) with minimal FPR. However, for lower imbalances ($r = 0.1, 0.3$), approaches like PTT-Random and Mid-Point yield high FPR values, reducing overall detection reliability. Notably, TabPFN FET-Mean and PTT-Majority are among the top-performing techniques, maintaining 100 % detection accuracy across all imbalance levels. The GMM baseline fall below 12 %, indicating random-like detection and MINE and PC-softmax detects more the 80 % of the ILs.

DamnVulnerableOpenSSL

Similar to OpenSSL-0.9.7a, for DamnVulnerableOpenSSL the MI-based AutoGluon Cal Log-Loss and TabPFN classification-based (FET-Mean, FET-Median, PTT-Majority) ILD approaches demonstrate strong detection rates of 90 %, especially in balanced datasets ($r = 0.5$), achieving near-perfect accuracy (100 %) with minimal FPR. However, imbalanced datasets ($r = 0.1, 0.3$) see elevated FPR for almost all techniques, specifically PTT-Random and Mid-Point, reaching 100 % FPR, which underscores their vulnerability in distinguishing false positives. Among the classification-based methods, FET-Mean, FET-Median, and PTT-Majority maintain robust accuracy levels, highlighting their adaptability across varying imbalance levels. The GMM baseline detect around 60 % of ILs, while MINE and PC-softmax falls below 50 %, , indicating random-like detection.

Cisco

The Cisco server, a vulnerable imitation, presents unique challenges in detection accuracy across all tested ILD approaches, with none surpassing an overall accuracy of 60 % on imbalanced datasets. Performance slightly improves on balanced datasets, as seen with TabPFN and AutoGluon methods like Mid-Point, FET-Mean, and PTT-Majority, which show consistency around 60 % accuracy. However, these methods still struggle with high false positive rate (FPR), especially in imbalanced scenarios, indicating significant sensitivity to class distribution shifts. This contrasts with the relatively better accuracy levels in other vulnerable servers, where most approaches detect 90 % of ILs.

Notably, while methods like PC-softmax achieve lower FPR in Cisco's balanced datasets, they are still hampered by false negative rates (FNRs) close to 100 % in imbalanced conditions, underscoring the limited generalization capability on this server. The consistently low accuracy across class imbalances highlights the Cisco server's distinct detection challenge, suggesting a need for specialized tuning to manage the unique IL signature.

Facebook

The Facebook server shows varied performance across ILD approaches, especially under different class imbalances.For high imbalances ($r = 0.1, 0.3$), TabPFN's IR Cal Log-Loss and HB Cal Log-Loss reach perfect accuracy with no false positives, while AutoGluon's PTT-Majority achieves 98 % accuracy with a minor false positive rate of 5 %.However, other approaches, particularly AutoGluon's Log-Loss and TS Cal Log-Loss, show moderate success, managing 86 % accuracy.With a balanced dataset, most techniques, including AutoGluon Mid-Point and TabPFN's FET-Mean, achieve 100 % accuracy, effectively reducing false positives and negatives, demonstrating a robust detection capability for Facebook's data patterns.

NetScalerGCM

On the NetScalerGCM server, ILD approaches illustrate differences in performance across imbalanced datasets. At $r = 0.1, 0.3$, AutoGluon's FET-Median attains perfect accuracy, while TabPFN's FET-Mean and PTT-Majority consistently achieve 100 % accuracy across all imbalances.This server highlights the effectiveness of both TabPFN and AutoGluon approaches in detecting information leakage, with TabPFN's IR Cal Log-Loss performing strongly as imbalance decreases. For balanced datasets, all approaches reach 100 % accuracy, implying high sensitivity of these techniques in identifying ILs.

PAN-OS

The PAN-OS server results show that both AutoGluon and TabPFN techniques can achieve near-perfect detection performance, especially on more balanced datasets. With high imbalance ($r = 0.1$), TabPFN's FET-Median and AutoGluon's FET-Mean reach 100 % accuracy, while others, like TabPFN's HB Cal Log-Loss, maintain zero false positives. As the dataset becomes more balanced, most approaches, including AutoGluon Log-Loss and TabPFN TS Cal Log-Loss, achieve 100 % accuracy. PAN-OS demonstrates high detection accuracy across different class imbalances in the dataset, making it one of the most detectable servers.

A.2.3. Summary

The results suggest that methods like FET-Mean, FET-Median, PTT-Majority, and calibration-based Cal Log-Loss approaches are well-suited for environments with significant vulnerabilities, consistently achieving high detection rates across servers vulnerable to error codes. Baseline methods, particularly MINE and PC-softmax, generally fail to detect vulnerabilities but occasionally perform well on non-vulnerable servers by correctly missing non-existent ILs, as noted in Sections A.4 and 4.3. This reflects their

tendency to underestimate MI, limiting their sensitivity to nuanced leaks in vulnerable contexts but proving useful for non-vulnerable scenarios. Notably, ILD approaches using AutoGluon outperform TabPFN on error code datasets, contrasting with TabPFN's superior performance on timing-based servers, as detailed in Appendix A.3 and Section 5.3. This suggests that AutoGluon excels in error code contexts, while TabPFN is better suited for timing-based vulnerabilities, emphasizing the importance of aligning AutoML tool selection with dataset characteristics.

A.3. ILD Generalizability on Timing Datasets

In this section, the generalizability of ILD approaches is explored in comparison to state-of-the-art methods. Heatmaps are employed to depict key performance metrics, such as Accuracy, FPR, and FNR, with respect to the time delay. The time delay is defined as the difference in computation time between correctly padded messages and manipulated (incorrectly) padded messages sent to the OpenSSL TLS server. The generalization capability of various ILD approaches to detect timing side channels is assessed to mitigate the Bleichenbacher attack, with a focus on performance metrics like detection accuracy, FPR, and FNR. The analysis uses the heatmaps with a fixed rejection threshold of 5 ($\tau = 5$) to illustrate the generalization performance of these ILD approaches. The performance is demonstrated relative to time delays in linear increments of 5 µs and to delays in logarithmic steps of 2 µs, in Figures A.2 and A.3, respectively.

A.3.1. TabPFN

This section delves into a comprehensive analysis of MI and classification-based ILD approaches using TabPFN. Typically, ILD approaches employing TabPFN show reduced effectiveness in detecting ILs in systems generating imbalanced

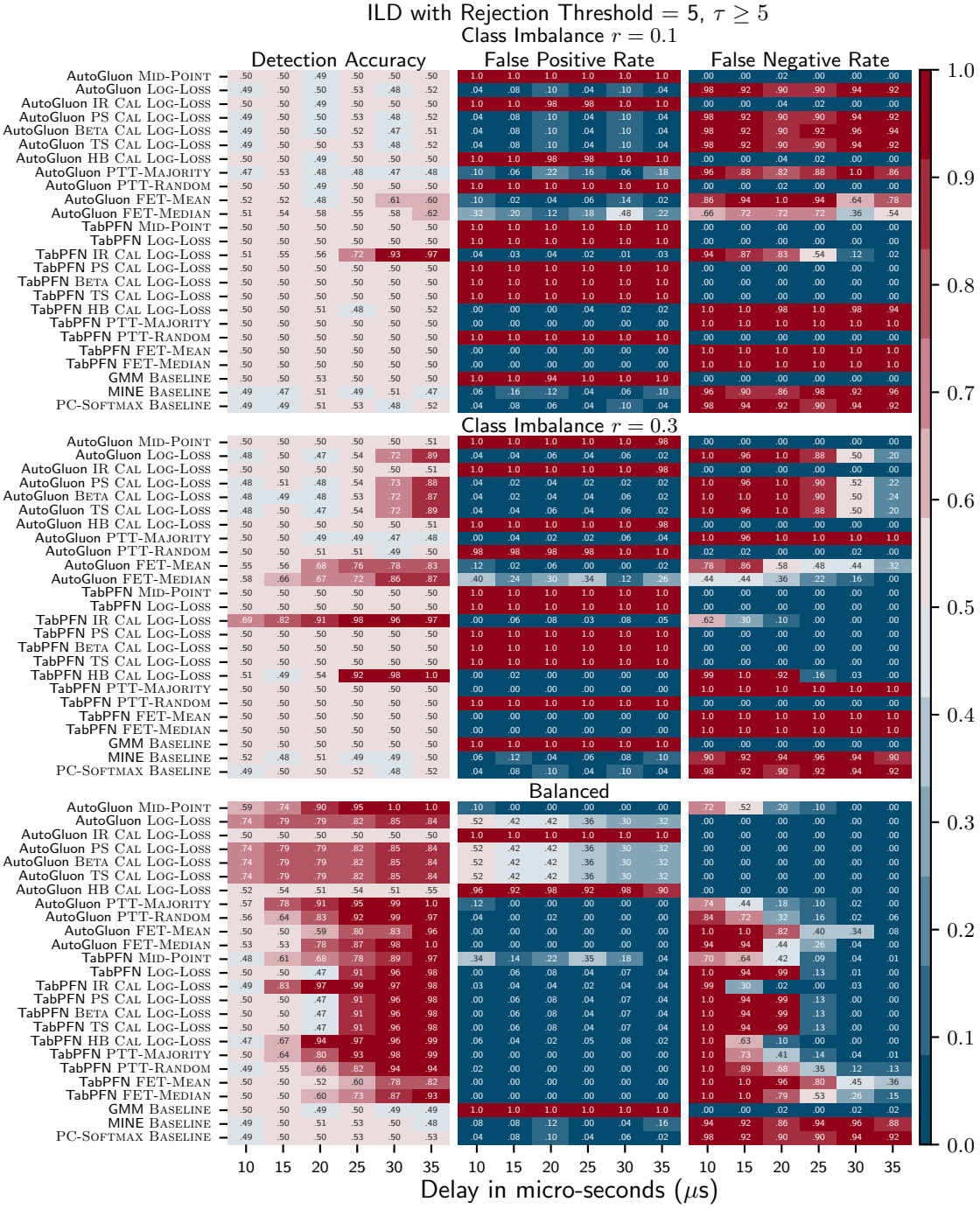


Figure A.2.: Performance of ILD approaches versus time delay with $5\,\mu s$ steps

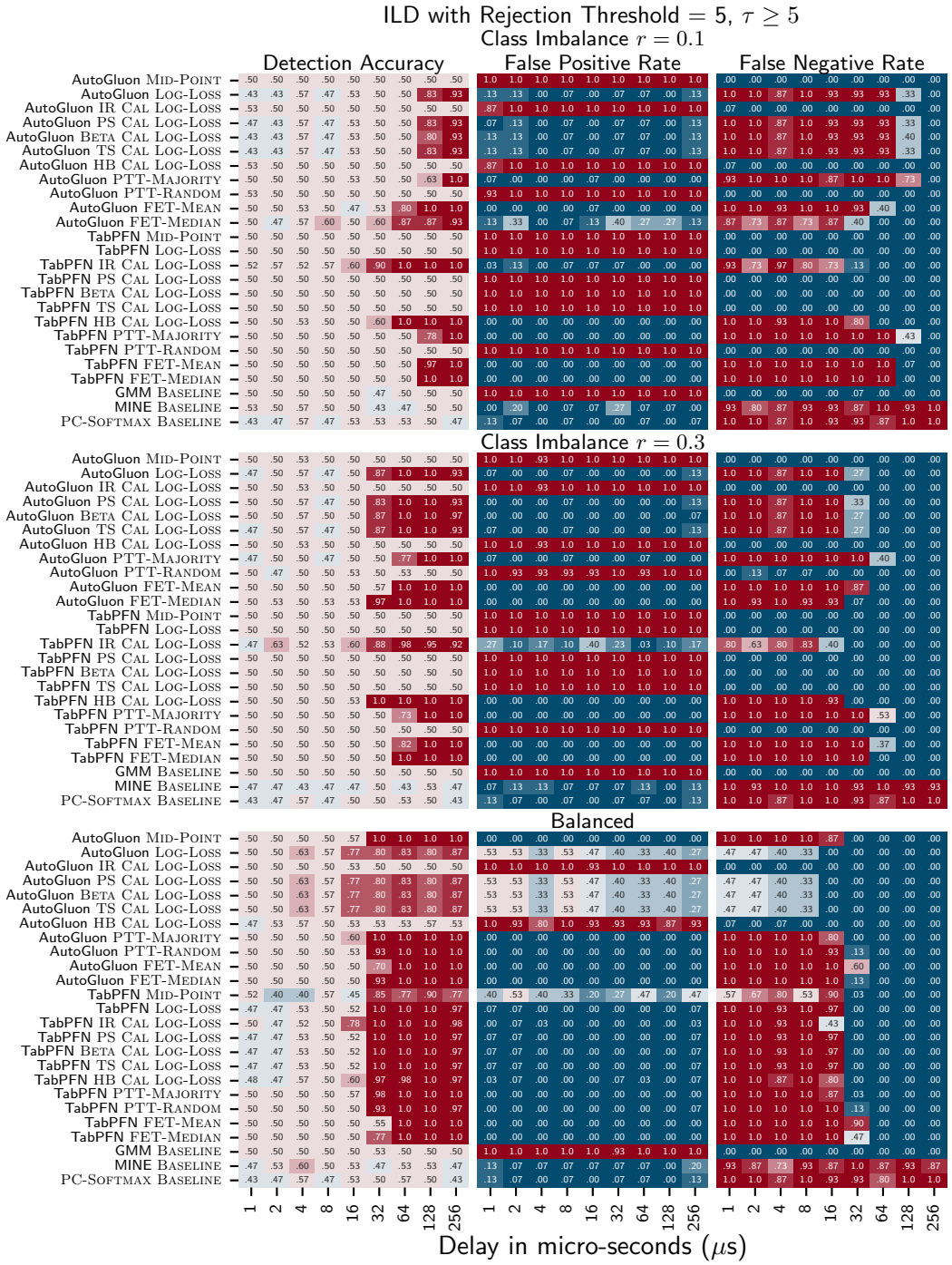


Figure A.3.: Performance of ILD approaches versus time delay with logarithmic step of $2\,\mu s$

datasets compared to those producing balanced ones. Remarkably, TabPFN IR Cal Log-Loss stands out in terms of generalization performance, detecting over 60 % of the ILs in systems with a minimal time delay less than 20 µs.

Classification-based Approaches

When focusing on the *balanced datasets*, TabPFN PTT-Majority stands out in detecting ILs. On the other hand, TabPFN FET-Median proves to be more adept in detecting ILs for *imbalanced datasets*. In the context of *imbalanced datasets*, the classification-based ILD approaches using TabPFN can detect ILs only when the time delay of the systems exceeds 35 µs. For delays shorter than 35 µs, these methodologies fall short in detecting in any ILs, resulting in a 100 % FNR. Amongst all, TabPFN FET-Median showcases the best detection performance for imbalanced synthetic datasets. Detecting ILs in systems generating *balanced datasets* is comparatively easier. In this context, the classification-based ILD approaches using TabPFN In systems producing *balanced datasets*, the classification-based ILD approaches using TabPFN can detect ILs in systems even with a slight time delay of at least 15 µs. For systems with minimal time delay below 15 µs, all these approaches fail in detecting ILs, yielding a 100 % FNR. Particularly, TabPFN PTT-Majority displays superior detection Accuracy, occasionally detecting over 60 % of the ILs in systems with time delay beyond 15 µs.

MI-based Approaches

Overall, TabPFN IR Cal Log-Loss is the most proficient in detecting ILs for balanced and imbalanced synthetic datasets. In the context of the *imbalanced datasets*, most MI-based approaches using TabPFN, except for IR Cal Log-Loss and HB Cal Log-Loss, face challenges in detecting 50 % of the ILs, leading to random detection decisions. This behavior arises because these approaches tend to overfit by overestimating MI, leading to the detection of non-existent ILs and yielding a 100 % FPR. Both TabPFN IR Cal Log-Loss

and HB Cal Log-Loss approach detect over 80 % of the ILs in systems with time delay above 32 µs. Impressively, TabPFN IR Cal Log-Loss approach occasionally detects over 55 % of the ILs in systems with very minimal time delay under 20 µs. This highlights the remarkable enhancement IR Cal Log-Loss brings to the Log-Loss approach, with HB Cal Log-Loss also contributing positively. For systems producing *balanced datasets*, MI-based ILD approaches using TabPFN display significantly improved performance. They can reliably detect ILs even in systems with short delays of 15 µs, with TabPFN IR Cal Log-Loss occasionally detecting more than 80 % of the ILs. However, for systems with delays under 15 µs, detection fails with a 100 % FNR.

A.3.2. AutoGluon

This section delves into a comprehensive analysis of classification and MI-based ILD approaches using AutoGluon. Generally, ILD approaches using AutoGluon face greater challenges with imbalanced synthetic datasets, especially with shorter time delays. Notably, the generalization capability of ILD approaches using AutoGluon is commendable, detecting occasionally over 65 % of ILs in systems, even with time delays under 20 µs.

Classification-based Approaches

The FET based classification ILD approaches using AutoGluon reveal comparable performances for both balanced and imbalanced synthetic datasets, albeit with minor differences. However, the AutoGluon PTT-Majority approach demonstrates superior detection performance on balanced synthetic datasets compared to the imbalanced ones. For *imbalanced datasets*, the classification-based ILD approaches using AutoGluon start reliably detecting ILs in systems with time delays exceeding 30 µs. Below this threshold, their detection performance plummets significantly, often missing the majority of ILs, resulting in an elevated FNR (typically around 100 %). The AutoGluon FET-Median

approach appears to have a slight edge over the other. When considering *balanced synthetic datasets*, AutoGluon's classification-based ILD approaches excel, reliably detecting ILs even in systems with a brief time delay of 15 µs, occasionally achieving over 60 % detection accuracy. Mirroring the findings from TabPFN, the PTT-Majority approach using AutoGluon consistently stands out as the top performer for balanced synthetic datasets.

MI-based Approaches

Upon a deeper performance inspection of MI-based ILD approaches using AutoGluon, it is evident that AutoGluon Mid-Point notably thrives in detecting ILs in balanced synthetic datasets, whereas AutoGluon Log-Loss is exceptionally proficient for imbalanced ones. Intriguingly, the calibration techniques (Cal Log-Loss) fail to enhance the performance of AutoGluon Log-Loss approach and, in some instances, even deteriorate its effectiveness. Within *imbalanced datasets*, AutoGluon's MI-based approaches generally struggle, especially for shorter delays. Their performance consistency improves for time delays beyond 25 µs, with Log-Loss variant emerging as the front runner. Most calibration techniques (Cal Log-Loss) do not augment the Log-Loss approach's performance. In fact, employing the IR Cal Log-Loss and HB Cal Log-Loss methods negatively impact its performance, leading to the detection of non-existent ILs, potentially stemming from overfitting or overestimating MI, leading to a 100 % FPR, which is consistent with the findings in Sections 4.3.1 and 5.3.1. As expected, the AutoGluon Mid-Point approach also mistakenly detects non-existent ILs, leading to a 100 % FPR. For *balanced datasets*, the MI-based approaches exhibit enhanced performance, with most methods detecting approximately 74 % ILs in systems with time delay over 10 µs, AutoGluon Mid-Point approach emerging as the front runner. However, like the imbalanced counterparts, calibration techniques (Cal Log-Loss) do not improve the Log-Loss approach's performance. In fact, employing the IR Cal Log-Loss and HB Cal Log-Loss methods diminish its efficacy, leading

to the detection of non-existent ILs, potentially stemming from overfitting or overestimating MI, leading to a 100 % FPR, which is consistent with the findings in Sections 4.3.1 and 5.3.1.

A.3.3. ILD Baselines

For the ILD baselines, the performance analysis predominantly indicates a consistent behavior, regardless of the system's datasets being **balanced** or **imbalanced**. Generally, all baselines consistently achieve a detection accuracy approximately equal to 50 %. Interestingly, while MINE and PC-Softmax often miss most of the ILs and produces false negatives, GMM leans towards the opposite end of the spectrum, generating false positives and overestimating MI. This tendency of GMM to overestimate MI is more pronounced in high-dimensional datasets, where it's traditionally known to be susceptible to misestimation due to overfitting. However, it's worth noting that occasionally, both MINE and PC-Softmax demonstrate better performance, detecting 60 % of ILs. Despite these occasional spikes in detection accuracy, the overall performance of baselines emphasizes the necessity and superiority of more specialized approaches in the realm of ILD.

A.3.4. Summary

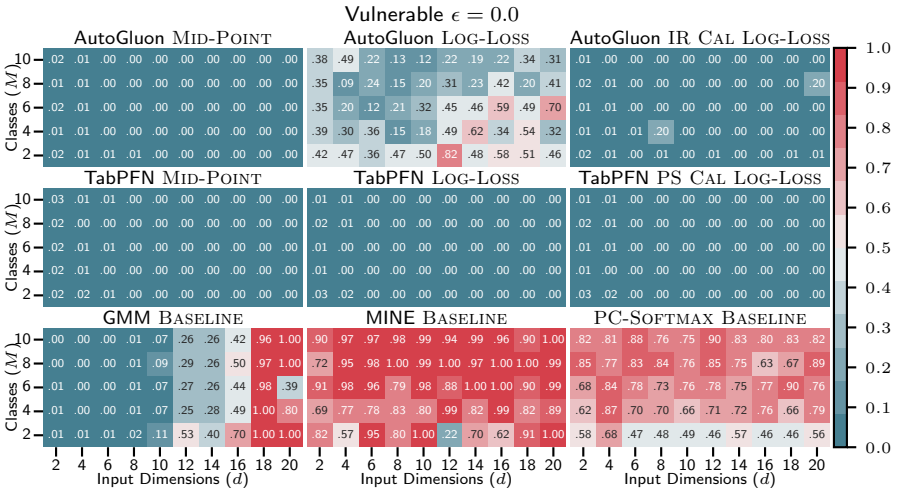
Summarizing the observations, while calibration techniques (Cal Log-Loss) tend to improve the detection capabilities of TabPFN Log-Loss, they sometimes either diminish the performance of AutoGluon Log-Loss or leave it unaffected. In contrast, calibration techniques (Cal Log-Loss) didn't significantly impact TabPFN Log-Loss's performance, as discussed in Section 4.3.1, which indicate that the application and necessity of calibration techniques are contingent upon the underlying system's datasets, as discussed in Section 5.3.3. For systems yielding balanced datasets, except IR Cal Log-Loss and HB Cal Log-Loss methods using AutoGluon when detecting nearly 50 % of ILs,

often errs by overlooking most ILs, resulting in an elevated FNR (typically around 100 %). When AutoGluon employs the IR Cal Log-Loss and HB Cal Log-Loss techniques detecting about 50 % of ILs, tends to make arbitrary detection decisions due to the identification of non-existent ILs, yielding a 100 % FPR, possibly because of overfitting or overestimation of MI. The observation in Sections A.4 and 4.3.1 back the reason behind this behavior, where the usage of calibration techniques for AutoGluon Log-Loss approach mostly leads to overfitting and overestimating MI in non-vulnerable synthetic datasets, resulting in the detection of non-existent ILs.

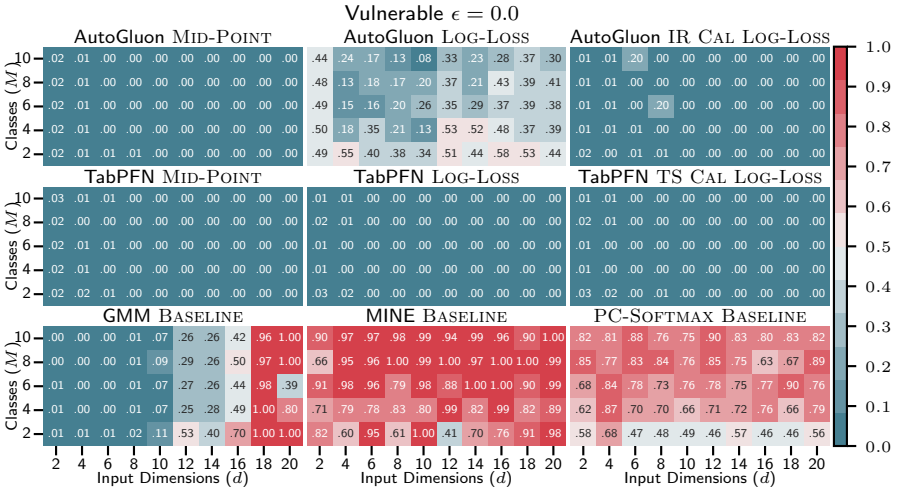
For systems with time delays between 1 to 256 μ-seconds that generate imbalanced datasets, the majority of MI-based approaches are prone to arbitrary detection decisions, producing approximately 50 % Accuracy. These approaches often misidentify non-existent ILs, incurring around a 100 % FPR, likely stemming from overfitting or MI overestimations. The remaining MI-based approaches, which detect over 50 % of the ILs in systems with longer time delays, generally falter due to missing new ILs and yielding high FNR. Similarly, classification-based approaches also underperform by failing to detect novel ILs and producing high FNR. The baseline ILD approaches remain subpar, merely detecting 50 % of the ILs across all systems, making random decisions due to elevated FPR or FNR (typically around 100 %).

In summary, overall TabPFN IR Cal Log-Loss consistently demonstrate robust performance across systems, whether those producing balanced or imbalanced datasets. When using AutoGluon, the FET-Median method stands out for imbalanced synthetic datasets, whereas the Mid-Point and PTT-Majority strategies excel at detecting ILs in the systems producing balanced datasets.

Notably, ILD approaches using TabPFN outperform AutoGluon on timing datasets, while AutoGluon excels on error code servers, as detailed in Appendix A.2 and Section 5.3. This suggests each AutoML tool is suited to specific vulnerabilities, reinforcing the importance of adapting tool deployment to the system's dataset characteristics.



(a) Balanced MVN perturbation synthetic dataset with $0\,\%$ noise level

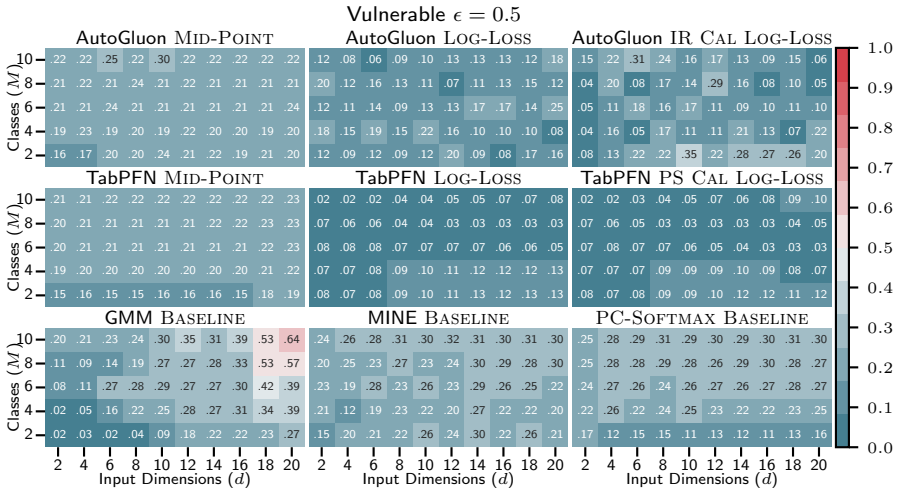


(b) Balanced MVN proximity synthetic dataset with $0\,\%$ noise level

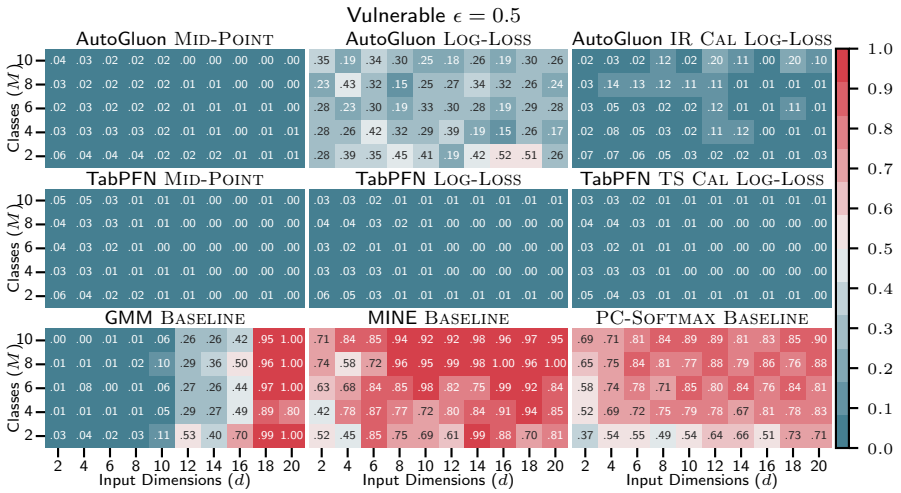
Figure A.4.: Generalizability of MI estimation methods on noise-free vulnerable systems

A.4. Generalizability of MI Estimation Methods

The generalization capability of the MI estimation methods is assessed using both AutoGluon and TabPFN in comparison to baseline methods, with the normalized mean absolute error (NMAE) performance metric defined in eq. (4.11). For each dataset type, namely balanced, binary-class, and multi-class imbal-



(a) Balanced MVN perturbation synthetic dataset with $50\,\%$ noise level

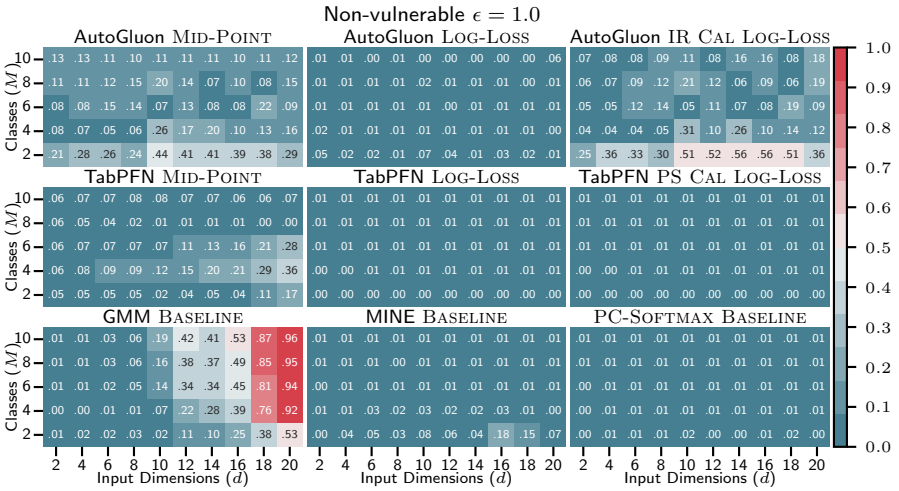


(b) Balanced MVN proximity synthetic dataset with $50\,\%$ noise level.

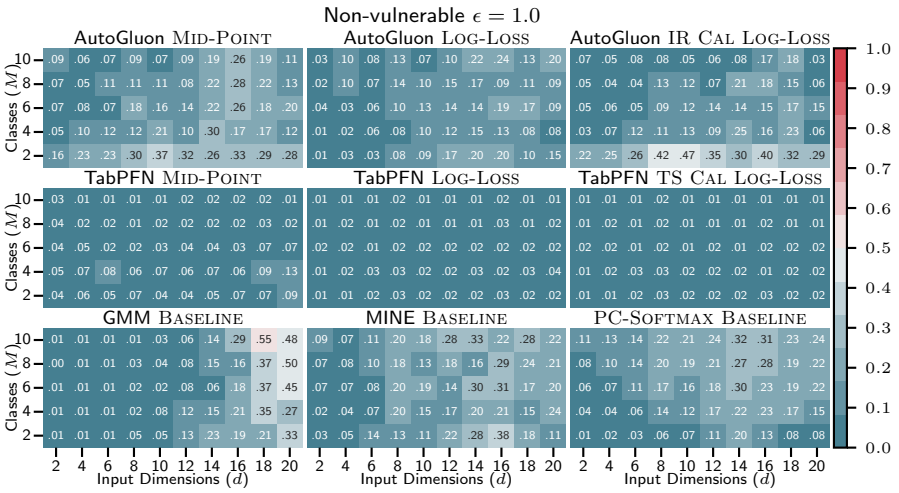
Figure A.5.: Generalizability of MI estimation methods on noisy vulnerable systems

ance, the NMAE is determined for every unique configuration of the number of classes (M), input dimensions (d), class imbalance (r), and noise level (ϵ). The top-performing calibration technique (selected from among IR Cal Log-Loss, PS Cal Log-Loss, Beta Cal Log-Loss, TS Cal Log-Loss, and HB Cal Log-Loss) is utilized to improve the efficiency of the Log-Loss approach for MI estimation due to spatial constraints. A comprehensive assessment is conducted, incorporating heatmaps to illustrate the generalizability of these

approaches. Specifically, the generalizability with respect to the number of classes (M) and input dimensions (d) in balanced synthetic datasets are discussed in Appendix A.4.1. Conversely, Appendix A.4.2 includes a discussion on the generalizability with respect to class imbalance (r) and noise level (ϵ) within binary-class and multi-class imbalance synthetic datasets.



(a) Balanced MVN perturbation synthetic dataset with 100 % noise level



(b) Balanced MVN proximity synthetic dataset with 100 % noise level

Figure A.6.: Generalizability of MI estimation methods on non-vulnerable systems

A.4.1. Number of Classes and Input Dimensions

To delve into a deeper understanding of the generalizability of MI estimation methods across balanced synthetic datasets, the NMAE performance is analyzed in terms of the number of classes (M) and input dimensions (d). In ***vulnerable systems*** at noise levels of 0 % and 50 % the performance results are depicted in Figures A.4 and A.5, respectively, while the performance on non-vulnerable synthetic systems at a noise level of 100 % are showcased in Figure A.6. The

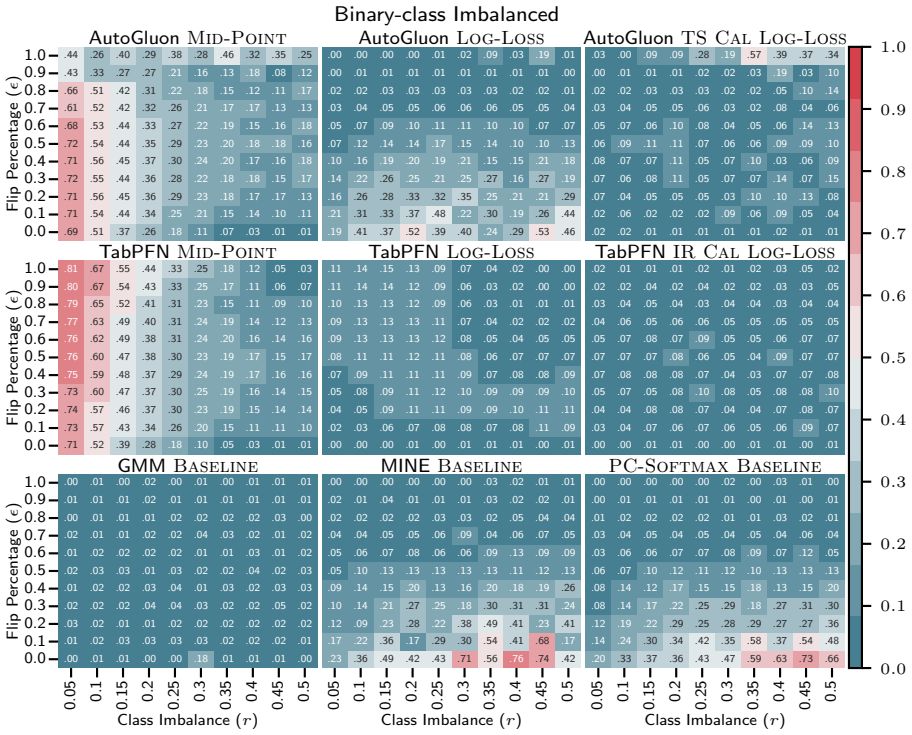


Figure A.7.: Generalizability of MI estimation methods on MVN perturbation synthetic binary-class imbalanced datasets

Y-axis represents the number of classes ($M \in \{2, 4, \ldots, 10\}$), and the X-axis represents input dimensions ($d \in \{2, 4, \ldots, 20\}$).

TabPFN

Overall, it is observed that the MI estimation methods using TabPFN, including Mid-Point, Log-Loss, and Cal Log-Loss, show strong generalization across classes (M) and dimensions (d). However, Mid-Point struggles for high-dimensional datasets ($d \geq 14$) simulated by ***vulnerable systems***, especially for MVN perturbation systems, reaching a maximum NMAE of 0.20. The TabPFN

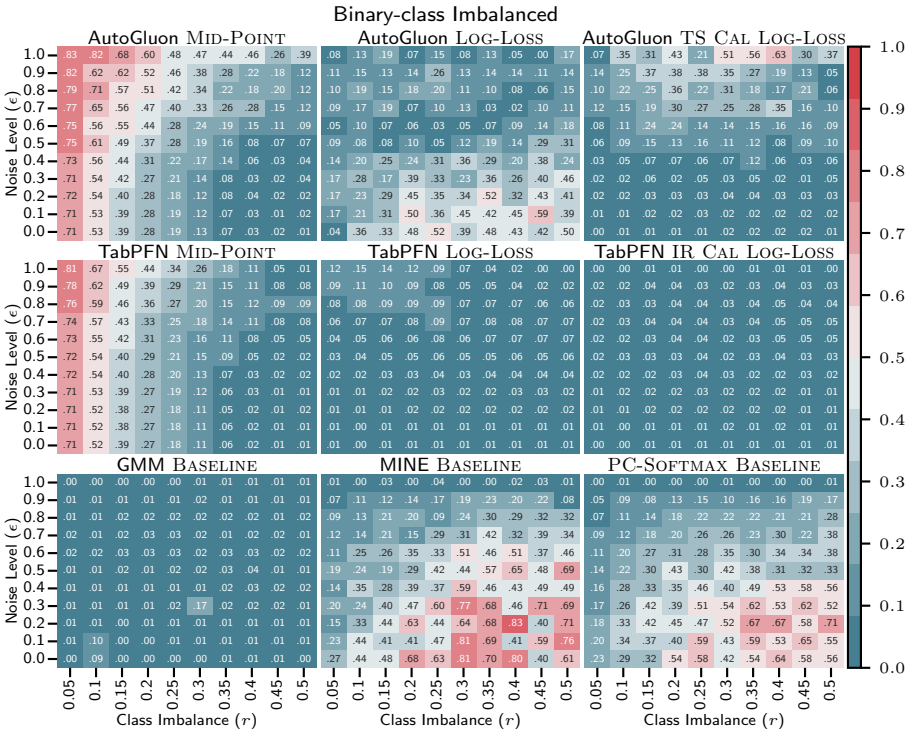


Figure A.8.: Generalizability of MI estimation methods on MVN proximity synthetic binary-class imbalanced datasets

Log-Loss and TabPFN Cal Log-Loss approaches consistently outperform the remaining, achieving an NMAE of around 0.01 across most cases, indicating estimation precision largely independent of the number of classes (M) and input dimensions (d), except in ***vulnerable systems*** with 50 % noise in MVN perturbation datasets Calibration (Cal Log-Loss) does not improve TabPFN Log-Loss precision, aligning with prior findings in Section 4.3.1 and consistent with the generalization results in Section 4.3.2.

AutoGluon

In most cases, all MI estimation methods using AutoGluon generally perform well across varying the number of classes (M) and input dimensions (d), with a few exceptional cases, which are missed in the findings in Section 4.3.2, due to loss of information due to aggregation. However, MID-POINT and CAL LOG-LOSS with AutoGluon tend to overestimate MI due to overfitting in ***non-vulnerable system*** with binary-class datasets ($M = 2$), resulting in false positives in ILD by detecting non-existent ILs, as also observed in Section 5.3. Additionally, CAL LOG-LOSS's performance with AutoGluon sometimes declines as the number of classes and input dimensions increases in datasets. Calibrated

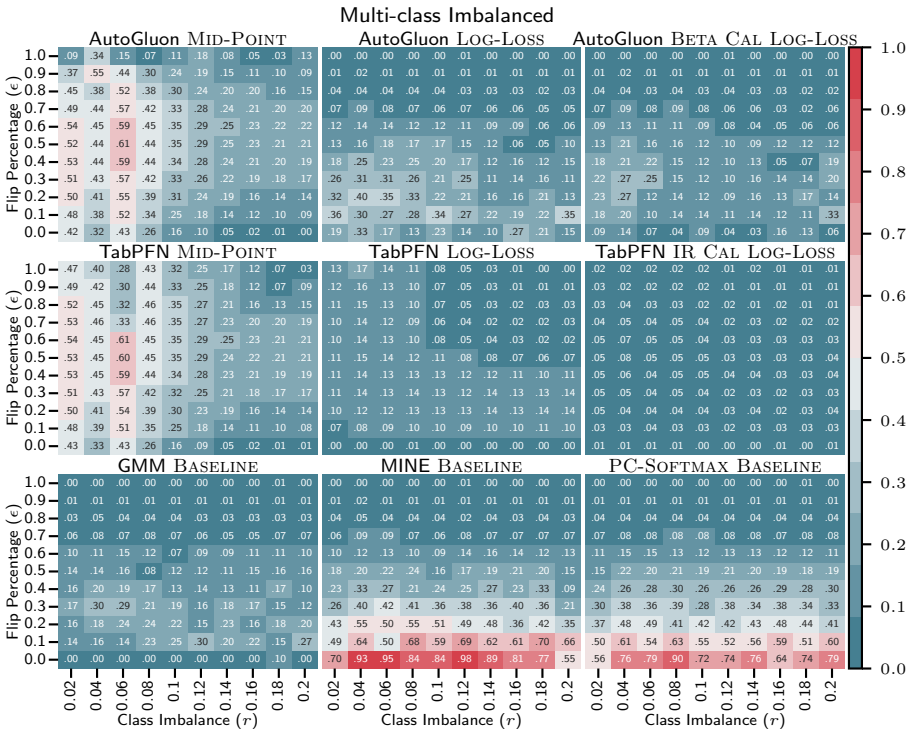


Figure A.9.: Generalizability of MI estimation methods on MVN perturbation synthetic multi-class imbalanced datasets

AutoGluon LOG-LOSS frequently overestimate MI in ***non-vulnerable systems***, while improving MI estimation precision in ***vulnerable systems***, as depicted

in Figure A.6 and Figures A.4 and A.5, respectively, aligning with the findings in Section 4.3.1. These observations on the generalizability of AutoGluon align with those in Section 4.3.2.

Baselines

All baseline approaches show reduced generalization for both **vulnerable** and **non-vulnerable systems**, with performance degrading as the number of classes (M) and input dimensions (d) increase. The GMM baseline especially

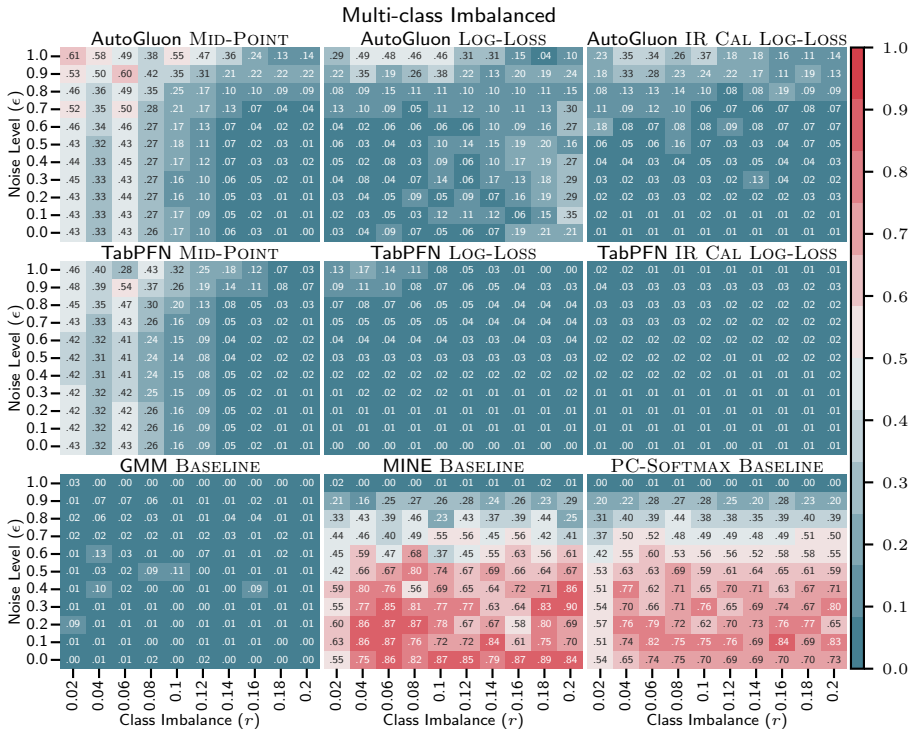


Figure A.10.: Generalizability of MI estimation methods on MVN proximity synthetic multi-class imbalanced datasets

struggles with high-dimensional data, degrading beyond 10 dimensions, yet generally surpasses PC-Softmax and MINE, except in non-vulnerable systems simulated with MVN perturbation, also observed in Sections 4.3.1 and 4.3.2.

A.4.2. Class Imbalance and Noise Level

The generalizability of MI estimation methods with respect to class imbalance (r) and noise level (ϵ) is analyzed by assessing their performance using NMAE on binary-class and multi-class imbalanced datasets generated through MVN perturbation and proximity techniques.The results are illustrated in heatmaps in Figures A.7 and A.8 for binary-class datasets and in Figures A.9 and A.10 for multi-class datasets. The Y-axis is represented by noise levels ($\epsilon \in [0.0, 1.0]$), while the X-axis corresponds to class imbalances, i.e., $r \in [0.05, 0.5]$ for binary-class datasets and $r \in [0.02, 0.2]$ for multi-class datasets.

TabPFN

Overall, Log-Loss and Cal Log-Loss approaches using TabPFN exhibit strong generalizability with respect to class imbalance and noise levels, showing resilience in both binary-class and multi-class imbalanced synthetic datasets. While TabPFN Mid-Point performs well regarding noise levels, it underperforms in highly imbalanced datasets due to its tendency to overestimate MI in these cases, as detailed in Section 3.3.1. Using calibration techniques (Cal Log-Loss) with TabPFN Log-Loss significantly enhances MI estimation precision, especially in multi-class systems simulated using the MVN perturbation technique. These findings align with the observations in Sections 4.3.1 and 4.3.2.

AutoGluon

The generalization of MI estimation methods using AutoGluon for class imbalances and noise levels in imbalanced datasets displays mixed results. The AutoGluon Mid-Point approach generally performs the worst, particularly in highly imbalanced datasets ($r \leq 0.2$ for binary-class and $r \leq 0.08$ for multi-class) and with increased noise levels, aligning with its tendency to overestimate MI in imbalanced cases, as discussed in Section 3.3.1. AutoGluon Log-Loss and

Cal Log-Loss approaches also tend to overestimate MI, showing overfitting under specific class imbalances and lower noise levels, especially in binary-class datasets generated by systems simulated using the MVN proximity technique. A distinctive observation with AutoGluon pertains to its response to changing noise levels. With increasing noise, Cal Log-Loss's performance fluctuates, revealing potential sensitivity to noise in imbalanced synthetic datasets. The AutoGluon Log-Loss approach with Cal Log-Loss performs well at average noise levels, demonstrating its utility in multi-class datasets, but calibration can reduce MI estimation accuracy in non-vulnerable, imbalanced datasets. These findings align with the observations in Sections 4.3.1 and 4.3.2.

Baselines

The MINE and PC-Softmax baselines generally struggle with imbalanced datasets, both binary and multi-class, as their generalization capability declines with decreasing class imbalance and noise levels. However, these baselines estimate MI accurately in binary-class and multi-class non-vulnerable datasets, also observed in Section 4.3. The GMM baseline generally adapts well to class imbalance and noise levels, especially in synthetic systems simulated using MVN perturbation, generating multi-class datasets, with slightly outperforming TabPFN in case of imbalanced binary-class datasets. The strong generalization of GMM is linked to the low dimensionality ($d = 5$) of imbalanced datasets, indicating its primary challenges lie in estimating MI for high-dimensional data, with noise and class imbalance having minimal impact. These observations on the generalizability of baselines align with the findings in Section 4.3.2.

A.4.3. Summary

In conclusion, TabPFN Cal Log-Loss consistently exhibits strong generalization in estimating MI across multiple factors in both MVN perturbation and proximity synthetic datasets. Notably, applying Cal Log-Loss calibration to

TabPFN Log-Loss significantly enhances MI estimation accuracy for vulnerable synthetic datasets but leads to considerable precision loss in non-vulnerable datasets. This disparity explains why the AutoGluon Cal Log-Loss ILD approach frequently detects non-existent ILs in OpenSSL TLS timing datasets, resulting in false positives, while TabPFN Cal Log-Loss surpasses other TabPFN-based ILD approaches, as seen in Sections A.3.4 and 5.3.1. However, the AutoGluon Cal Log-Loss ILD technique performs effectively in detecting ILs in OpenSSL TLS error code datasets, as highlighted in Section 5.3.2. Overall, baseline methods struggle with high-dimensional, imbalanced, noisy synthetic datasets, with these limitations becoming especially evident when contrasted with Cal Log-Loss approaches using TabPFN and AutoGluon. In OpenSSL TLS error code datasets, for instance, GMM detects under 50 %, performing below random guessing, whereas MINE and PC-softmax exceed 62 %, which contrasts with their performance on timing datasets (see Section 5.3.1). These observations align with the findings in Section 4.3.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

00FPB 0x00 `In PKCS#1 Padding (First 8 Bytes After` 0x00 0x02`)`.

00LP 0x00 `On The Last Position (`$|PMS| = 0$`)`.

00NLP 0x00 `On The Next To Last Position (`$|PMS| = 1$`)`.

00SPB 0x00 `In Some PKCS#1 Padding Byte`.

AES Advanced Encryption Standard.

AIC Akaike information criterion.

AutoML automated machine learning.

AutoSCA Automatisierte Schwachstellenanalyse von kryptographischen Protokollen.

BCE binary cross-entropy.

BER balanced error-rate.

BMBF Bundesministerium für Bildung und Forschung.

BO Bayesian optimization.

BOHB Bayesian optimization and Hyperband.

BS Brier score.

CASH combined algorithm selection and hyperparameter optimization.

CatBoost categorical boosting machine.

CBC cipher block changing.

CCE categorical cross-entropy.

CCS ChangeCipherSpec.

CFPM `Correctly Formatted PKCS#1 Message`.

CFPM1BS `Correctly Formatted PKCS#1 PMS Message But 1 Byte Shorter`.

CFPM47 `Correctly Formatted PKCS#1 Message` $|PMS| = 47$.

CKE ClientKeyExchange.

CM confusion matrix.

CNN convolutional neural network.

DL deep learning.

DL-LA Deep Learning Leakage Assessment.

DPA differential power analysis.

DV Donsker-Varadhan.

EM expectation-maximization.

EMR electromagnetic radiations.

EMV Europay-Mastercard-Visa.

ERC European Research Council.

FET Fisher's exact test.

FIN Finished.

FNR false negative rate.

FPR false positive rate.

GBM gradient boosting machine.

GE guessing entropy.

GMM Gaussian mixture model.

HB Hyperband.

HD hamming distance.

HDDC high-dimensional data clustering.

HPO hyperparameter optimization.

HT hardware Trojan.

HTM hierarchical temporal memory.

HW hamming weight.

ID identity.

IKE Internet Key Exchange.

IL information leakage.

ILD information leakage detection.

IoT internet of things.

ITV `Invalid TLS Version In PMS`.

IV initialization vector.

JSSE Java Secure Socket Extension.

KDE Kernel density estimation.

KFCV K-fold cross-validation.

KL Kullback-Leibler.

KNN K-nearest neighbor.

KPR Klíma-Pokorný-Rosa.

LAS leakage assessment score.

LightGBM light gradient boosting machine.

MAE mean absolute error.

MCC mathews correlation coefficient.

MCCV Monte Carlo cross-validation.

MI mutual information.

MINE mutual information neural estimation.

MitM man-in-the-middle.

ML machine learning.

MLP multi-layer perceptron.

MSE mean squared error.

MVN multivariate normal.

N00M `No` 0x00 `In Message`.

NAS neural architecture search.

NEMIA Neural Estimated Mutual Information Analysis.

NMAE normalized mean absolute error.

NN neural network.

OAEP optimal asymmetric encryption padding.

OTT one-sample t-test.

OVR one-versus-rest.

PAVA pool adjacent violators algorithm.

PC-softmax probabilitiy-corrected softmax.

PDF probability density function.

PMF probability mass function.

PMS pre-master secret.

POI Point of Interest.

PQKE post-quantum key-exchange.

PS padding string.

PTT paired t-test.

RF random forest classifier.

ROC Receiver Operating Characteristic.

RSA Rivest–Shamir–Adleman.

RSM rotating Sbox masking.

RST `reset`.

SASCA Soft analytical side-channel attacks.

SCA side-channel attack.

SE standard error.

SNR signal-to-noise ratio.

SR success rate.

SSL Secure Sockets Layer.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TPM trusted platform module.

TST trace sufficiency threshold.

VPN virtual private network.

VS vulnerability score.

WFB `Wrong First Byte` (0x00 `Set To` 0x17).

WSB `Wrong Second Byte` (0x02 `Set To` 0x17).

XGBoost eXtreme gradient boosting machine.

XT extra trees classifier.

Declaration of Authorship

I hereby declare that this thesis is my original work, independently composed without unauthorized materials or unacknowledged assistance. All sources and aids used have been appropriately cited or acknowledged. This dissertation has not been previously submitted to any other faculty or institution. Additionally, I confirm that I have not undergone an unsuccessful doctoral examination nor been stripped of any previously earned postgraduate (Dr.) degrees. Any text passages quoted verbatim or closely paraphrased from external sources and figures taken from or adapted from external sources are explicitly marked and adequately referenced. Figures based on foundational concepts or significantly modified are not referenced in the caption.

I further declare that this thesis has not been submitted to any other examination board in the same or similar form.

....

Place, Date Pritha Gupta