PriVA-C: Defending Voice Assistants from Fingerprinting Attacks

Dilawer Ahmed
North Carolina State University
Raleigh, NC, USA
dahmed2@ncsu.edu

Aafaq Sabir
North Carolina State University
Raleigh, NC, USA
asabir2@ncsu.edu

Ahsan Zafar
North Carolina State University
Raleigh, NC, USA
azafar2@ncsu.edu

Anupam Das
North Carolina State University
Raleigh, North Carolina
anupam.das@ncsu.edu

Abstract

Voice assistants have become ubiquitous, yet they remain vulnerable to network traffic fingerprinting attacks that can expose sensitive user information. Existing defenses either impose high overheads or fail against advanced attacks. This paper addresses these issues by introducing and evaluating PriVA-C, a fingerprinting defense mechanism tailored specifically for voice assistants. Unlike prior approaches that treat voice assistant traffic as generic web traffic, we analyze its unique characteristics to design a more effective defense. Our approach prioritizes limiting information leakage rather than targeting specific attack vectors, achieving a significant reduction in attacker accuracy from 89% to 13%. We also propose a more practically deployable version of our defense, which protects only traffic directed to the primary voice assistant domain, reducing attacker accuracy to 19%. We implement a functional prototype using the Alexa SDK, conduct user testing, and assess its performance using real network traffic. Our results demonstrate that our proposed defense effectively mitigates fingerprinting attacks while maintaining low overhead and preserving the user experience.

Keywords

Traffic Analysis; Privacy; Fingerprinting, Voice Assistants

1 Introduction

Voice assistants have become increasingly popular as an alternative mode of interaction [43]. Recent trends and projections indicate an increasing presence of voice assistant-enabled smart speakers in homes, along with growing trends in voice-based searches [27]. The convenience of asking for information through your voice rather than typing a search query makes the search faster and more convenient for users for multiple common tasks such as asking general questions, getting weather updates, playing music, etc. With Alexa's recent upgrade to incorporate Large Language Model (LLM) capabilities for native and developer use, this trend is expected to continue growing [7].

Recent studies have highlighted various privacy concerns on voice assistant platforms, including unauthorized invocations [20], issues with unwanted data collection by third-party skills [32, 48], and unintended ad profiling [28]. Another significant privacy threat arises from the analysis of encrypted network traffic to identify patterns in user commands issued to voice assistants [4, 11, 24, 30, 31, 36]. By detecting these patterns, attackers can infer sensitive user activities, revealing personal details such as daily routines, interests, and preferences. Previous studies have demonstrated the effectiveness of these attacks across various platforms and command types, highlighting the privacy risks to end users.

While prior research has presented a few proof-of-concept countermeasures to defeat such fingerprinting attack [24, 59], we find these techniques to be ineffective against more sophisticated attacks. Other works [13, 21] have proposed platform-agnostic countermeasures that operate at the network communication level, but these solutions are often too generic and introduce significant usability issues. This limits their effectiveness in achieving meaningful reductions in attack performance while remaining practical for real-world deployment. Additionally, some of these countermeasures require modifications to the network or routing layers, making them impractical to implement across millions of already deployed devices in the wild.

In this work, we seek to answer the following research questions. **RQ1:** *Are current countermeasures for voice command and website fingerprinting effective in preventing voice command fingerprinting attacks with reasonable overheads*? This allows us to understand any inefficiencies, limitations, and unrealistic assumptions that existing works have. **RQ2:** *Based on insights from analyzing voice assistant traffic, how can we design a more privacy-preserving and efficient countermeasure that effectively balances privacy protection with utility and system overhead?* We introduce "PriVA-C" (pronounced *privacy*) as a practical and effective defense against voice command fingerprinting. Finally, **RQ3:** *How can we practically implement our proposed defense against voice command fingerprinting?* Most proposed countermeasures require anonymous routing to protect all voice assistant traffic and in typical home networks, without a VPN, this assumption fails. We propose and evaluate a version of PriVA-C only applied to traffic to only one endpoint which removes the need for proxies or VPNs to make deployment more realistic. To the best of our knowledge, we are the first to develop a working prototype of this countermeasure on the Alexa SDK and to conduct both system performance analysis and usability testing with users.

Our approach draws inspiration from prior research on traffic analysis defenses, particularly in the context of website and webpage fingerprinting [29, 34]. It capitalizes on the unique patterns

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–16
© YYYY Copyright held by the owner/author(s).
https://doi.org/XXXXXXX.XXXXXXX

in how voice assistant devices communicate with networks in response to user commands. By selectively obscuring specific traffic characteristics during different phases of voice assistant communication , our method anonymizes both the user's command and the server's response while reducing the overhead of simplistic constant-rate defense approaches. Specifically, we build on fixed-rate defenses by adapting them across different phases, dynamically adjusting traffic rate characteristics to optimize overhead. Additionally, we leverage data-driven analysis to select optimal transmission rates, minimizing both delay and communication overhead. Our design also incorporates protection for traffic directed to the primary voice assistant domain, enhancing the overall deployability of the solution. We parameterize our defense to allow users a balance between overhead and privacy. Through a user study, we show that this defense is practical for real-world deployment and does not degrade the user experience. Our key research contributions are:

- We conducted a comprehensive evaluation of existing defenses for voice assistant and website fingerprinting, highlighting their strengths and limitations in mitigating voice command fingerprinting attacks.
- We designed and implemented, PriVA-C, a voice assistant fingerprinting defense, inspired by existing countermeasures. We identified optimal configurations tailored to the unique characteristics of voice assistants, enabling PriVA-C to maintain strong performance while adapting multiple fingerprinting attack vectors and voice assistant platforms.
- We integrated our defense into the Alexa SDK and analyzed real-world data to rigorously quantify our defense's effectiveness, offering valuable insights into the challenges and opportunities in this domain.
- We also conduct a user study with our defense integrated into Alexa SDK to assess its impact on user experience and system performance.
- To foster further research and innovation, we open-source our code and dataset[1], allowing researchers and developers to replicate our results, extend our work, and contribute to the advancement of voice assistant privacy.

2 Background

2.1 Network Traffic-Based Fingerprinting

Network fingerprinting involves identifying unique characteristics in network traffic or device behavior to distinguish one entity from another. The core concept is that each target—whether a device, software, or app—has distinct characteristics, use cases, and requirements, leading to unique communication patterns. By analyzing encrypted traffic, these patterns can be classified and attributed to specific devices, often through machine learning techniques. This approach is applied across various domains [8, 18, 38, 57], each with its own methodologies and implications. Network fingerprinting can serve as both a privacy attack—such as identifying the operating system of a device [42]—and a defense mechanism, like detecting rogue devices on a network [2]. Website fingerprinting, IoT fingerprinting, and voice assistant fingerprinting are sub-domains of network fingerprinting.

[1]https://github.com/dilawer11/PriVA-C

Website Fingerprinting. Website fingerprinting involves identifying visited websites by analyzing traffic patterns, such as data volume, packet timing, and sequences [15, 25, 46, 54]. Over time, this technique has become increasingly sophisticated, even capable of identifying webpages accessed simultaneously on the Tor network. Despite Tor's encryption, anonymous routing, and privacy features, website fingerprinting can still compromise user privacy by revealing browsing habits. Researchers have developed countermeasures to mitigate such attacks, emphasizing the delicate balance between security and privacy [37].

IoT Fingerprinting. IoT fingerprinting identifies devices based on their unique communication patterns, including packet frequency, size, and protocols. Researchers have applied this technique for various purposes, such as inventory management, anomaly detection, and privacy attacks, particularly by identifying devices within home networks [9, 44, 49]. Studies have examined how fingerprints evolve over time and how factors such as geo-location affect IoT device identification [3]. These attacks can even reveal specific activities performed on IoT devices, such as motion detection, raising further privacy concerns [9]. Additionally, other unique identifiers can be used to pinpoint specific devices or systems [16, 50].

Voice Command Fingerprinting. Voice command fingerprinting is a subset of IoT fingerprinting that focuses specifically on identifying voice assistant devices and their activities or commands. This technique can recognize specific commands, such as asking a voice assistant to *play music* or inquire about the *weather*. Privacy attacks using this method can target multiple platforms, including Amazon Alexa, Google Assistant, and Siri [4, 59]. While previous research [24, 59] has proposed methods to secure this traffic, we will evaluate these existing solutions, highlight their limitations, and suggest improvements to better protect user privacy.

Each type of fingerprinting underlines the dual-edged nature of network analysis, offering benefits while posing privacy and security challenges. Understanding these aspects is crucial for developing robust defense mechanisms against potential threats.

2.2 Threat Model

In a voice command fingerprinting attack, the threat model considers the bidirectional network traffic between a voice assistant device and the internet endpoints it communicates with to provide functionality. The device is assumed to be deployed in a typical home network that offers standard encrypted internet connectivity but lacks additional protections against traffic monitoring (e.g., a VPN). The user interacts with the assistant by issuing voice commands, while an on-path adversary passively observes the resulting traffic. Such an adversary could be non-local, for instance, an ISP. Due to NAT, non-local adversaries cannot directly attribute an observed network flow to its originating device within the home network. Consequently, traffic from multiple active devices may become intermingled, introducing noise into the target device's flows and potentially degrading feature extraction. To address this challenge, Ahmed et al. [4] proposed a flow filtering technique that isolates the traffic of voice assistant devices from that of other devices, demonstrating that attacks can still achieve high accuracy even in noisy conditions. An alternative threat model involves a local adversary with direct access to the home network. Such an adversary—e.g., a roommate—can easily associate traffic with its originating device and is therefore considered more powerful than a non-local adversary. There may also be a disparity in capabilities between an ISP, which could possess significantly greater computational and financial resources, and a roommate, whose resources may be limited. Nevertheless, such an attack can be carried out with modest means, such as a consumer-grade laptop. In either setup, the attacker can monitor metadata—such as packet sizes, timing, and frequency—for encrypted traffic, and may also access the contents of unencrypted traffic, such as DNS queries. By leveraging the knowledge of any deployed countermeasures and configurations, the attacker can build a fingerprint database or train models to infer user commands, i.e., the attacker can adapt strategies based on the deployed countermeasures. The attacker through this attack can infer user behavior and commands without decrypting the communication, thereby compromising privacy.

3 Related Works

While research on network traffic analysis and privacy preservation is extensive, this section focuses on the most relevant areas: IoT fingerprinting, voice command fingerprinting, and website fingerprinting, which form a foundation for understanding and addressing the unique challenges of voice command fingerprinting defenses.

3.1 Fingerprinting Attacks

There is extensive research on fingerprinting devices and applications through traffic analysis [2, 15, 18, 25, 38, 46, 52, 54, 55, 57]. Among these, the most relevant to our work are studies focused on IoT fingerprinting and voice command fingerprinting.

IoT Fingerprinting. IoT fingerprinting focuses on identifying devices based on their network traffic and has been widely explored in prior research. This body of work demonstrates the feasibility of extracting meaningful information from network patterns. Techniques range from applying machine learning and deep learning to network metadata [9, 33, 49, 56], to analyzing specific elements such as DNS queries [44]. Additionally, researchers have investigated the factors that enable fingerprinting under diverse network conditions [3, 16].

Voice Command Fingerprinting. Voice command fingerprinting poses a direct threat to user privacy. Initial work by Kennedy et al. [30] demonstrated the feasibility of this attack on Alexa. Subsequent research has sought to improve accuracy [36], extend analysis to open-world scenarios [59], incorporate multiple voice assistants [4], and explore the impact of location-based factors [31]. Researchers have also shown the effectiveness of such an attack in the presence of a Home VPN [24]. These studies collectively demonstrate the increasing sophistication and real-world applicability of voice command fingerprinting attacks.

3.2 Fingerprinting Defenses

Although defenses against voice command fingerprinting are still in their early stages, the broader domain of network traffic analysis countermeasures — especially within website fingerprinting — provides valuable insights. These defense strategies generally fall into several key categories:

Fixed-Rate Transmission: Early defenses such as BuFLO [14], Tamaraw [13], and CS-BuFLO [12] sought to counter fingerprinting attacks by enforcing fixed transmission rates. Although these approaches proved effective in reducing attack success, they introduced substantial overhead, making them impractical for real-world use. To improve efficiency, DynaFlow [34] introduced a more adaptive strategy by dynamically adjusting transmission rates based on traffic conditions.

Anonymity Sets: Defenses like Glove [41] and Supersequence [60] group sensitive websites into anonymity sets, aiming to make traffic within each set indistinguishable. Walkie-Talkie [61] was built upon this concept. However, these methods often run into various problems during implementation [37, 46], which can be unrealistic for resource-constrained IoT devices.

Padding-Based Approaches: Techniques like WTF-PAD [29] and BiMorphing [5] introduce padding to obfuscate traffic patterns. However, recent research suggests these methods often suffer from high overhead or limited effectiveness [54].

Adversarial Examples: Methods like Mockingbird [45] and BANP [40] generate specific traffic patterns to disrupt machine learning-based attacks. However, their effectiveness is often limited to specific attack methods [37].

Traffic Splitting: This approach splits traffic across multiple paths e.g., Tor circuits [19] or ISPs [26]. While conceptually effective, it often requires infrastructure not readily available in typical home network settings.

Voice-platform Specific: Research on concrete defenses against voice command fingerprinting is still limited. Wang et al. [59] proposed a padding-based method, inspired by WTF-PAD, to defend against their own deep learning-based fingerprinting attack. Another recent effort, VoiceDefense [24], also addresses the challenge of protecting against voice command fingerprinting by padding and fake traffic injection.

3.3 Distinction with Prior Work

We significantly differ from previous efforts of voice command fingerprinting defenses by Wang et al. [59] and Guo et al. [24]. Wang et al. extended Adaptive Padding, similar to WTF-PAD, and introduced delays via differential privacy. However, this approach suffers from substantial time overheads and proves ineffective against more sophisticated attacks that leverage packet timing information. Guo et al.'s method, which employs random reshaping of outgoing traffic and the addition of invalid incoming requests, is also vulnerable to timing-based attacks, as it fails to mitigate the information leakage from packet timing as we show in Section 4.

In contrast, our research focuses on fundamentally limiting distinctive traffic patterns, rather than targeting a specific attack strategy. While this general-purpose approach may introduce higher overhead, prior research [13, 37] and our own findings in Section 4 demonstrate that it offers stronger and more resilient privacy protection. Inspired by constant-rate defenses [13, 21], we have optimized these techniques specifically for voice assistants, achieving a more efficient and robust solution. Furthermore, we are the first to not only present a functional defense prototype but also to conduct usability testing and evaluation using real network traffic. This practical, empirical approach, coupled with our rigorous evaluation

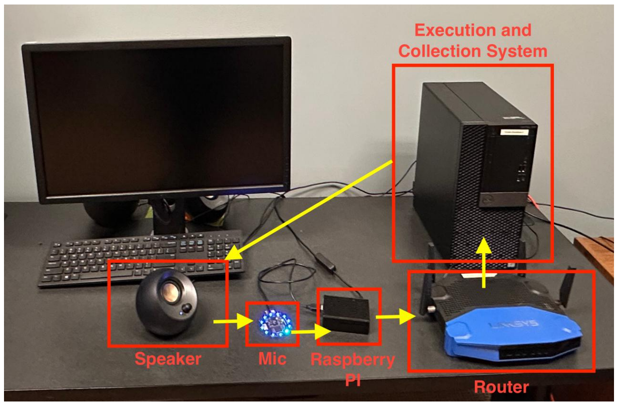


Figure 1: The lab deployed a setup to collect data, where the control system plays audio request commands through speakers. These commands are picked up by a microphone connected to a Raspberry Pi running the AVS SDK. Network traffic is captured at the router that provides internet connectivity to the Raspberry Pi.

against multiple publicly available attacks, demonstrates the broad applicability and effectiveness of our defense, setting it apart from the theoretical or proof-of-concept nature of prior voice command fingerprinting defense studies.

4 Evaluating Existing Defenses

In this section, we describe the experimental setups and methodologies used to evaluate the effectiveness and limitations of existing defenses against voice command fingerprinting (**RQ1**).

4.1 Collection Setup & Datasets

Amazon Alexa is the most widely used voice assistant in terms of market share [22] and existing literature on voice command fingerprinting [4, 11, 24, 30, 31, 59] has primarily focused on Alexa. Amazon Alexa provides an open-source SDK [6] to integrate the voice assistant functionality into third-party devices. For the aforementioned reasons we primarily focus on Amazon Alexa throughout the remainder of this paper since we use the Alexa SDK to implement, test, and deploy our defense in Section 5. However, we do evaluate existing approaches and our defense on the second and third most popular voice assistant platforms, Google and Siri, respectively [22], to demonstrate adaptability. However, for these two platforms, we relied on data collected in prior work [4], as neither Google nor Siri provides an open-source SDK that would allow us to modify their existing communication patterns.

In line with previous work [24, 37, 54, 59], we simulate the application of countermeasures on network traffic to evaluate their performance and any overhead they may introduce to transmission. We utilize the simulation code provided by the authors where available, following the instructions in their papers as closely as possible. Our dataset encompasses a diverse set of command types—simple queries, streaming requests, and third-party skill invocations—each generating distinct network traces. Such diversity is essential for building a benchmark dataset capable of evaluating countermeasure performance across varied scenarios, as different command types may exhibit unique traffic characteristics [4]. The resulting *benchmark dataset* consists of 100 voice commands, each repeated 100 times, yielding a total of 10,000 samples. To ensure that the Alexa SDK consistently responded with the correct output, we manually validated one complete round of data collection (i.e., for all 100 different voice commands) and reused the same set of invocation audio files for the remaining 99 rounds. To collect the dataset, we developed an automated data collection pipeline based on the methodologies used in prior works [4, 59], ensuring consistency with prior research. The data collection setup is visually shown in Figure 1. It consists of the following items:

- Amazon Alexa SDK [6] running on a Raspberry Pi 4 (4 GB) with a UMA-8 USB Microphone Array as the voice assistant device.
- The voice assistant device is connected via Wi-Fi to an OpenWRT Linksys router, with network traffic captured and stored on an Ubuntu Desktop for increased storage capacity.
- A speaker, controlled by a Python script, automatically converts the text command to audio using a Text-to-Speech model and plays it, thereby activating the voice assistant.

For Google Assistant and Siri, we use the open-source datasets provided by Ahmed et al. [4]. Each dataset contains 5,000 samples, comprising 100 recordings of 50 distinct simple voice commands, collected using a setup similar to that employed for the Alexa SDK.

4.2 Baseline Attack Models

We use two recent open-sourced voice command fingerprinting attacks from the latest literature [4, 59] to establish a baseline for attack performance on unprotected, raw network traffic from voice assistants.

DeepVC [59]. DeepVC, proposed by Wang et al., is a voice command fingerprinting attack that utilizes various Deep Learning models, such as CNN and LSTM. The model architecture is inspired by website fingerprinting attacks, like Deep Fingerprinting [54]. The attack focuses on packet size and direction features, where each sample consists of the size and direction of the first n packets in the packet trace. For our evaluation, we experimented with both the default parameters recommended in their study and a range of hyperparameter tuning configurations, ultimately selecting the optimized settings that yielded the best performance. Although the datasets we evaluate differ from those used in their work, we were able to replicate their attack's performance on their datasets within an average 2% negative margin in performance metrics.

Spying [4]. Ahmed et al. introduced a novel voice command fingerprinting attack that utilizes multiple handcrafted features and automated machine learning pipelines to train the models. Each sample in their feature set is a summary of various metadata distributions and counts within a variable — size window. In Ahmed et al.'s [4] work they use AutoGluon Tabular [10] and Random Forest as the classifiers. AutoGluon trains a diverse ensemble of models and typically outperforms Random Forest by at most 5%. In our replication using Random Forest, attack accuracy for Google Assistant and Siri remained within 1% of results from AutoGluon, given identical datasets and feature parameters. For Alexa, AutoGluon achieved 4% higher accuracy. Despite this, we use Random Forest with default settings in our evaluations, as it enables efficient comparison across thousands of parameter combinations during grid

Table 1: Fingerprinting performance across three different platforms. The results show high attack accuracy in all cases for Spying attack [4] while performance for DeepVC [59] is higher for Google Assistant compared to other platforms.

Platform	Attack	Accuracy	Precision	Recall
Alexa	Spying	88.40%	85.83%	86.02%
	DeepVC	54.03%	59.40%	54.84%
Google Assistant	Spying	88.40%	88.49%	88.29%
	DeepVC	81.30%	86.04%	81.17%
Siri	Spying	85.90%	85.78%	86.20%
	DeepVC	64.90%	69.46%	65.16%

search for all defenses. However, we evaluated the performance of AutoGluon against the final defense (with optimized parameters) proposed by Ahmed et al. [4], and compared it with our Random Forest model to demonstrate that the defense remains effective regardless of the classifier employed by the attack. The corresponding results are reported in Tables 13 and 14 in Appendix A.

We partition the benchmark dataset using a standard 80:20 train/test split. For baseline performance evaluation, we use the raw network metadata without applying any defenses or modifications, reflecting typical voice assistant communication patterns. This setup aligns with the methodology adopted in prior attack studies. The results are presented in Table 1. Ahmed et al. [4] outperform Wang et al. [59] in fingerprinting undefended traffic, achieving an accuracy of 88% compared to 54%. We also test the respective models on Google Assistant and Siri. We achieve similar results with high undefended accuracy for both platforms showcasing the need for an effective countermeasure against such attacks.

4.3 Evaluation Setup

We test multiple traffic analysis countermeasures designed for both website fingerprinting and voice command fingerprinting. Each countermeasure is typically parameterized to accommodate various types of network traffic and operational scenarios. In our evaluation, we analyze each countermeasure across different parameter settings, at least as expansive as suggested by the original work.

It is crucial to highlight that the majority of website fingerprinting attacks and defenses have been studied within the context of the Tor network. The Tor network inherently provides anonymous tunneled routing and has the ability to transmit data in fixed-size chunks, known as cells. Consequently, many of these defenses are tailored to this environment and, in addition to not obfuscating destination addresses, assume each packet (cell) is padded to a fixed size. For our evaluations, we adapt these defenses, where applicable, to a fixed-packet size configuration as to not create a disadvantage for such defenses.

To assess the impact of countermeasures, we utilize bandwidth overhead and delay alongside standard classification performance metrics, including Accuracy, Precision, and Recall. Bandwidth overhead refers to the additional amount of data transmitted due to the countermeasure, including any padding or dummy traffic. The bandwidth overhead is calculated using the following percentage increase formula:

$$B/W_{overhead} = \frac{V_{\text{final}} - V_{\text{initial}}}{V_{\text{initial}}} \times 100 \tag{1}$$

where V_{initial} represents the original traffic volume, whereas V_{final} represents the total volume after applying the countermeasure.

Delay is a measure of time overhead, which is defined as the difference between the time that a packet was originally supposed to be sent (in undefended traffic) compared to the time it was actually sent after the defense was applied.

$$Delay = \frac{1}{n}\sum_{i=1}^{n}(T_{\text{defended}} - T_{\text{plain}}) \tag{2}$$

where n represents the total number of packets in the trace. If a packet is split into multiple parts and these parts arrive at different times, $T_{defended}$ is the time when the last part arrives (T_{plain} refers to the original timestamp of the packet). The delay metric excludes processing or transmission delays, as they are assumed to remain consistent and cancel out.

Given that our threat model assumes a fully informed adversary with complete knowledge of the deployed countermeasure and its parameters, we adopt a conservative evaluation approach. Specifically, we assume a worst-case scenario in which the adversary acts last and is able to train their models on traffic that has already been processed by the defense mechanism. Accordingly, in all our evaluations, we apply the countermeasure to both the training and test subsets of the benchmark dataset, thereby ensuring that the adversary operates under maximally favorable conditions.

4.4 Voice Command Fingerprinting Defenses

DeepVC-CM [59] proposed a proof-of-concept defense against their own voice command fingerprinting attack, DeepVC. The defense works by generating dummy packets based on the adaptive padding principle, sampling from a distribution of intervals and sizes of real packets. A Laplacian noise function dictates the size of both real and dummy packets as well as any delays to be incorporated. We use the open-sourced implementation provided to evaluate this defense. We found that differential privacy noise function occasionally causes unrealistically significant packet delays, due to highly varied packet sizes from different domains.

Our evaluation of DeepVC-CM [59] on Alexa (Table 2) shows that its most robust configuration reduces attacker accuracy to approximately 37%, but at the cost of a 131% bandwidth overhead. In the lowest bandwidth setting, accuracy increase to 42%. We note that we get abnormally high delays, thousands of seconds, due to a combination of large packet sizes observed in streaming commands, Laplacian noise design, and how DeepVC's simulation code applies the delays [58, 59]. Reducing delay by adjusting the Laplacian noise function is possible, however, would likely weaken its ability to conceal timing features. These results indicate that, although the defense effectively obfuscates traffic, its high latency limits real-world viability. However, for Google Assistant and Siri, we do not observe similarly large delays. This is likely due to the absence of large packet sizes in their datasets, which consist solely of simple commands (i.e., no streaming commands), resulting in comparatively lower delays on these platforms relative to Alexa. Nonetheless, when compared to other defenses, the delays remain significantly higher, indicating the need to re-evaluate and adapt the noise function to accommodate multiple scenarios rather than being tailored to a single dataset or attack.

Table 2: Attack performance after applying existing countermeasures on all traffic from the voice assistant using the benchmark dataset.

Countermeasure Details			Performance			Overheads	
Name	**Setting**	**Attack**	**Accuracy**	**Precision**	**Recall**	**B/W**	**Delay(s)**
DeepVC-CM [59]	Strongest	Spying	37.42%	36.22%	37.41%	131%	6323
		DeepVC	0.95%	0.44%	1.00%	411%	1061
	Low B/W	Spying	41.62%	40.93%	41.35%	94%	8956
		DeepVC	2.20%	1.99%	2.09%	94%	8956
VoiceDefense [24]	Strongest	Spying	77.09%	77.16%	77.26%	66%	0
		DeepVC	4.00%	6.08%	4.74%	12%	0
	Low B/W	Spying	82.59%	82.83%	82.70%	12%	0
		DeepVC	4.00%	6.08%	4.74%	12%	0
BuFLO [21]	Strongest	Spying	9.80%	11.99%	9.97%	1003%	0.45
		DeepVC	2.05%	0.57%	2.39%	645%	0.76
	Low B/W	Spying	37.02%	36.04%	36.78%	196%	1.78
		DeepVC	24.36%	26.11%	24.67%	196%	1.78
Tamaraw [13]	Strongest	Spying	16.41%	10.07%	16.88%	330%	0.81
		DeepVC	13.56%	9.47%	14.17%	220%	1.78
	Low B/W	Spying	23.56%	17.81%	24.22%	145%	2.53
		DeepVC	19.21%	10.41%	19.90%	145%	2.53
DynaFlow [34]	Strongest	Spying	11.66%	4.63%	12.26%	196%	4.11
		DeepVC	0.65%	0.01%	1.00%	196%	4.11
	Low B/W	Spying	11.66%	4.63%	12.26%	196%	4.11
		DeepVC	0.65%	0.01%	1.00%	196%	4.11
WTF-PAD [29]	Strongest	Spying	69.83%	69.87%	69.92%	84%	0
		DeepVC	0.70%	0.16%	0.69%	84%	0
	Low B/W	Spying	78.74%	79.37%	79.01%	9%	0
		DeepVC	1.40%	1.30%	1.38%	9%	0

VoiceDefense [24] is a proof-of-concept defense developed to mitigate voice command fingerprinting attacks. This approach introduces "random reshaping" of outgoing traffic and proposes blending incoming traffic with "invalid requests". However, the specifics of these invalid requests are not detailed, and the available implementation from the open-source repository does not include the code for invalid requests, with only the random reshaping component provided. Based on the paper's description, we interpret "invalid requests" as instances where the voice assistant is activated without any follow-up command, causing it to listen for a limited time without issuing a response. We use this assumption to simulate and evaluate the proposed defense in our study.

Our evaluation of VoiceDefense [24] on Alexa (as shown in Table 2) shows that even in its strongest configuration, it only reduces attack accuracy to 77%, with a bandwidth overhead of 66% and no delays. The lowest bandwidth configuration performs slightly worse, with an accuracy of 83% and a bandwidth overhead of 12%. These results indicate that while the approach is lightweight, it does not provide strong privacy guarantees against fingerprinting attacks. Similar ineffectiveness as a defense is also observed when evaluated in Google Assistant and Siri (as highlighted in Table 3) with the lowest accuracy in Google Assistant of 68%.

4.5 Website Fingerprinting Defenses

BuFLO [21]. Dyer et al. [21] argue that efficient countermeasures inherently leak information and are therefore ineffective. To address this, they propose a bi-directional constant-rate padding scheme, known as BuFLO, as an intentionally inefficient yet robust defense against website fingerprinting. BuFLO transmits traffic at a *constant rate* (e.g., every 20 ms) using *fixed-size packets* (e.g., 500 bytes) for at least a specified *minimum duration* which causes BuFLO to incur significant bandwidth overhead but guarantee strong performance.

Our evaluation shows that BuFLO, in its strongest configuration, reduces attack accuracy for Alexa to 10% (Table 2), but at the cost of sending 1003% more data. A lower bandwidth configuration of 645% overhead reduces the attacker accuracy to 37%. On Google Assistant and Siri a reduced attacker accuracy of about 2% (as shown in Table 3) is observed in both cases with similarly high overheads. These results underscore the trade-off between effectiveness and resource consumption, rendering BuFLO impractical for real-world deployment.

Tamaraw [13]. Tamaraw, proposed by Cai et al., improves upon BuFLO by introducing direction-specific transmission rates, acknowledging that web traffic typically involves more incoming than outgoing data [13]. Additionally, Tamaraw replaces BuFLO's minimum time parameter with a padding parameter to determine the defense's end condition. Instead of stopping immediately after the actual transmission ends, Tamaraw continues to send dummy packets until the total duration aligns with a multiple of the padding parameters. This obscures the precise end time of the transmission, further mitigating information leakage to an adversary.

Our results show that Tamaraw's strongest configuration reduces attack accuracy to 16%, with a bandwidth overhead of 330% and a moderate delay of 1.13 seconds. In its lowest bandwidth setting, accuracy rises to 24%, while overhead decreases to 145%. Analysis across platforms (as shown in Table 3) shows better results on Google Assistant and Siri with an accuracy of 4% and 9% respectively with similar overheads. These results suggest that Tamaraw offers a more practical balance between effectiveness and efficiency compared to BuFLO.

DynaFlow [34]. Lu et al. propose DynaFlow, an adjustable constant-rate defense similar to Tamaraw [13]. DynaFlow periodically modifies the transmission rate based on prior inter-packet delays and sends packets in fixed patterns at direction-specific rates to reflect differences in incoming and outgoing traffic. While its design aims to reduce overhead through adaptive rate adjustments, the parameters provided are tailored for website fingerprinting and may not directly translate to voice command fingerprinting due to differing traffic characteristics. We evaluate DynaFlow using both the parameters suggested in the original work and additional settings tailored to voice command traffic.

Our evaluation shows that DynaFlow's strongest and lowest-bandwidth configuration was the same in our parameter sweep, reducing attacker accuracy to 12% with an overhead of 196% at a mean latency of 4.11 seconds. These results indicate that while DynaFlow offers lower overhead than BuFLO, it still introduces significant latency due to its design tailored to website fingerprinting traces [34]. DynaFlow's performance worsens for Siri compared to other platforms (as shown in Table 3).

WTF-PAD [29]. Juarez et al. propose a probabilistic link-padding defense, extending the concept of Adaptive Padding [53]. This defense introduces dummy packets during periods of low network traffic to disrupt the characteristic patterns used by machine learning models for fingerprinting. WTF-PAD leverages the natural fluctuations in traffic streams, alternating between bursts of packets and quiet intervals. While it proved effective against earlier website fingerprinting attacks relying solely on packet sizes and

Table 3: Evaluating different countermeasures on Google Assistant and Siri. Results show a high overhead and/or low privacy performance of existing countermeasures.

Countermeasure Details			Google Assistant					Siri				
			Performance			Overheads		Performance			Overheads	
Name	Setting	Attack	Accuracy	Precision	Recall	B/W	Delay(s)	Accuracy	Precision	Recall	B/W	Delay(s)
DeepVC-CM [59]	Strongest	Spying	26.60%	24.82%	26.18%	144%	8.09	33.63%	31.11%	33.20%	556%	0.52
		DeepVC	2.30%	0.05%	2.00%	106%	4.99	2.00%	0.04%	2.00%	556%	0.52
	Low B/W	Spying	31.60%	28.61%	31.47%	106%	4.99	44.24%	44.54%	43.65%	53%	11.44
		DeepVC	2.30%	0.05%	2.00%	106%	4.99	25.33%	32.38%	25.82%	53%	11.44
VoiceDefense [24]	Strongest	Spying	67.70%	66.73%	67.22%	73%	0.0	76.58%	76.42%	75.97%	498%	0.0
		DeepVC	57.50%	70.90%	57.90%	69%	0.0	1.70%	0.43%	2.08%	429%	0.0
	Low B/W	Spying	79.40%	79.26%	79.29%	22%	0.0	81.38%	81.36%	80.85%	105%	0.0
		DeepVC	72.80%	79.82%	72.97%	22%	0.0	62.96%	71.80%	63.76%	105%	0.0
BuFLO [21]	Strongest	Spying	1.60%	0.28%	1.68%	1063%	0.12	1.70%	0.61%	1.83%	1169%	0.02
		DeepVC	1.60%	0.06%	1.84%	785%	0.21	1.70%	0.03%	2.00%	503%	0.04
	Low B/W	Spying	10.90%	10.68%	11.04%	400%	0.46	44.74%	44.13%	44.37%	411%	0.04
		DeepVC	3.90%	3.03%	4.10%	400%	0.46	2.00%	0.04%	2.00%	411%	0.04
Tamaraw [13]	Strongest	Spying	4.10%	0.57%	4.79%	322%	0.72	8.91%	2.00%	11.52%	344%	0.06
		DeepVC	1.40%	0.06%	2.18%	898%	0.12	1.60%	0.03%	2.00%	493%	0.04
	Low B/W	Spying	7.30%	3.28%	7.96%	302%	0.72	20.62%	13.60%	22.87%	321%	0.06
		DeepVC	8.00%	3.55%	7.72%	302%	0.72	2.00%	0.04%	2.00%	321%	0.06
DynaFlow [34]	Strongest	Spying	4.30%	1.44%	5.06%	1492%	0.41	21.82%	16.53%	24.26%	328%	0.44
		DeepVC	1.10%	0.02%	2.00%	1466%	0.8	2.00%	0.04%	2.00%	439%	0.07
	Low B/W	Spying	6.60%	3.09%	7.84%	347%	3.74	21.82%	16.53%	24.26%	328%	0.44
		DeepVC	1.60%	0.03%	2.00%	347%	3.74	5.71%	2.33%	5.71%	328%	0.44
WTF-PAD [29]	Strongest	Spying	66.10%	65.84%	66.06%	89%	0.0	72.97%	72.81%	72.26%	136%	0.0
		DeepVC	1.80%	0.04%	2.00%	64%	0.0	25.83%	32.85%	25.97%	136%	0.0
	Low B/W	Spying	79.90%	80.29%	80.15%	9%	0.0	76.88%	77.52%	76.43%	14%	0.0
		DeepVC	25.70%	35.24%	26.73%	9%	0.0	39.84%	50.14%	40.52%	14%	0.0

directions, more advanced attacks incorporating timing information have bypassed WTF-PAD, as it does not conceal the original timing of packets [46].

Our evaluation (Table 2) shows that WTF-PAD's strongest configuration reduces the accuracy to 70%, with a modest bandwidth overhead of 84% and no added delay. The lowest bandwidth setting causes an accuracy jump to 79% while reducing bandwidth usage to 9%. We observe similar trends on Google Assistant and Siri (as highlighted in Table 3). These results indicate that while WTF-PAD is lightweight, it does not provide strong privacy protection against fingerprinting attacks.

Other defenses. Researchers have proposed various countermeasures to prevent website fingerprinting on Tor, with traffic-splitting defenses [26] being a promising low-overhead, high-performance solution in specific threat models. However, in most home networks, it is unrealistic to assume the presence of multiple ISPs for voice assistant devices. Additionally, countermeasures that rely on adversarial examples to defeat specific attack vectors are ineffective against other attack models [21, 37], and as such, we do not consider them in this work. Other defenses are either challenging to implement for voice assistants [61] or require large databases [5], which are not feasible for low-powered IoT devices.

Takeaway. Existing voice command fingerprinting defenses are largely ineffective against diverse attack types. While website traffic fingerprinting techniques—such as constant rate defenses—show promise, they are not optimized for the unique characteristics of voice application traffic.

5 PriVA-C

In this section, we present the design and evaluation of multiple setups of PriVA-C to address **RQ2** & **RQ3**. As highlighted by previous works and our analysis of defenses discussed in Section 4, constant-rate defenses are most effective at obscuring network patterns and provide robust privacy against various traffic analysis attacks. However, they tend to incur higher overhead compared to other defenses, making optimization of these defenses crucial. We identify several key areas for potential improvement.

5.1 Traffic Pattern Analysis & Defense Design

To develop a defense mechanism that ensures privacy while minimizing overhead, it is crucial to first understand the traffic patterns of voice assistant interactions. This understanding helps identify the information that needs to be concealed from potential adversaries, while also identifying opportunities to reduce overhead. To achieve this, we conducted an in-depth analysis of the traffic patterns associated with 100 different voice assistant commands from our benchmark dataset. Our analysis uncovers several key insights:

Incoming vs. Outgoing. To assess the sensitivity of each traffic direction to fingerprinting, we divided the dataset into two parts. The first dataset contains only outgoing traffic, while the second includes only incoming traffic. We then extracted features from each dataset and trained the model. The results in Table 4 reveal that an adversary can fingerprint voice commands with approximately 79% accuracy using only outgoing traffic and 78% accuracy with only incoming traffic. This highlights the importance of securing traffic in both directions to achieve substantial privacy protection for the

Table 4: Fingerprinting performance using only incoming or outgoing traffic in Alexa. This shows the importance of defended bidirectional traffic.

Direction	Attack	Accuracy	Precision	Recall
Outgoing	Spying	78.89%	78.83%	79.02%
	DeepVC	38.82%	40.93%	39.20%
Incoming	Spying	77.99%	77.68%	78.00%
	DeepVC	46.15%	52.14%	47.74%

user. Modifications to outgoing traffic must be implemented on the "client-side" (before an adversary can observe outgoing traffic), while changes to incoming traffic should be made on the server side. The "server-side" refers to any service or point beyond which the adversary cannot intercept the traffic. This may include VPN services, reverse proxies, or even the endpoint itself.

Contacted Endpoints. Figure 7 in Appendix A demonstrates that approximately 65% of the commands in our benchmark dataset interact with a single domain, the Alexa Voice Service (AVS) Endpoint. For most of the remaining commands, the endpoints contacted are enough information to uniquely identify them; for instance, the command to play CNBC news connects to a subdomain of `cnbc.com`. In such cases, a tunneled proxy would be necessary to prevent information leakage from endpoints, requiring the voice assistant platform to route all traffic through itself, or the user to use a tunneled proxy. This tunneled setup can be complex to set up for an average user and/or increase the costs of operation for the platform owner. In our design, we consider two deployment scenarios. First, in which all the traffic is tunneled and protected, and another in which only traffic to the AVS domain is protected. We expect the former to be more privacy-preserving, however, the latter is more realistically deployable by the platform owner.

Outgoing Phase. Once the wake word is detected, the voice assistant begins streaming encrypted user audio from the microphone to the Amazon Alexa AVS Endpoint [1, 4]. This phase of communication is represented by the blue bars in Figure 2. The duration of this phase reflects how long it takes for the user to complete the command or for the Alexa Voice Service to interpret the user's intent. Except for the duration, this phase is similar for all command types. While the outgoing phase alone does not provide enough information to infer the user's voice commands — since factors like user age, speaking speed, and accent can affect the audio stream [59] — we propose a design that extends the duration of the outgoing phase. This extension can be achieved either by adding dummy packets at the same rate as the original traffic or by keeping the audio stream active. In our design, all outgoing traffic phases last for at least *out_min_len* seconds. Any outgoing phase longer than this is padded until the next interval, where the interval is defined by *outgoing_interval*. For instance, if *out_min_len = 5* and *outgoing_interval = 2*, an outgoing phase lasting 2 seconds will be padded to appear as 5 seconds to a network observer, while an outgoing phase of 6 seconds will be padded to 7 seconds.

Incoming Phase. Immediately, after the initial outgoing phase we observe a more incoming heavy traffic phase. This aligns with voice assistants' response to the user. This incoming phase exhibits a high degree of similarity for different instances of the same command

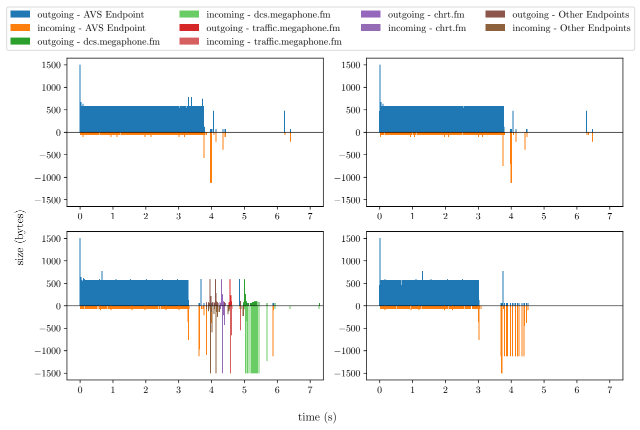


Figure 2: Network traffic for different commands. Positive sizes refers to outgoing traffic and negative sizes refers to incoming traffic (from the client's perspective). The network traffic has been adjusted to show a maximum of 1500 bytes for presentation.

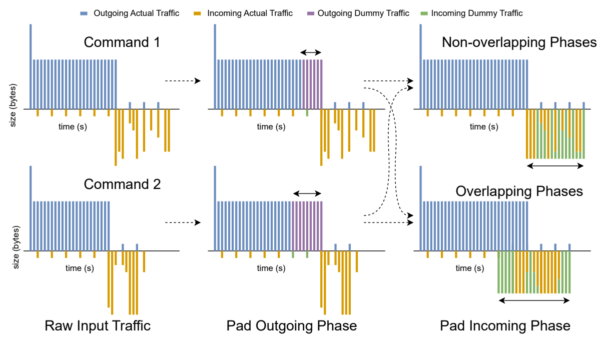


Figure 3: PriVA-C operates by independently padding each phase of communication. Both the outgoing and incoming phase are padded to the next interval to obscure their duration and potential revealing traffic patterns. Depending on the configuration, incoming and outgoing phases may overlap—resulting in lower latency but higher traffic volume—or remain non-overlapping, which reduces traffic volume at the cost of increased delay.

but varies across different command and command types. The start timing of this incoming phase can leak information about the true duration of the outgoing phase. One way to avoid this is to postpone the start of this phase, however, this increases the response latency and could negatively impact the user experience. Alternatively, the incoming phase can "appear" to start earlier by inserting dummy traffic before the incoming phase begins. We add a delay, controlled by parameter *in_delay* after which the incoming phase begins from the start of the outgoing phase. Increasing the value of this parameter reduces the bandwidth overhead but increases the latency and vice versa. Alternatively *in_delay* can be set to *null* to make the incoming phase start after the "padded" outgoing phase ends. The incoming phase terminates after a fixed duration defined by parameter *initial_max_len*.

Table 5: Different configurable parameters of PriVA-C with corresponding units.

Parameter	Description (unit)	Haven	Forge	Delta	Strong
out_fast_rate	The rate of outgoing packets when outgoing phase is active during initial transmission (pkts/s)	50	50	50	50
out_slow_rate	The rate of outgoing packets when outgoing phase is inactive during initial transmission (pkts/s)	4	4	4	4
out_ext_rate	The rate of outgoing packets during the extended transmission (pkts/s)	50	50	50	50
in_fast_rate	The rate of incoming packets when incoming phase is active during initial transmission (pkts/s)	50	50	50	50
in_slow_rate	The rate of incoming packets when incoming phase is inactive during initial transmission (pkts/s)	4	4	4	4
in_ext_rate	The rate of incoming packets during the extended transmission (pkts/s)	50	50	50	50
out_fast_size	The fixed size of outgoing packets when outgoing phase is active (bytes)	577	577	577	577
out_slow_size	The fixed size of outgoing packets when outgoing phase is inactive (bytes)	250	250	250	250
out_ext_size	The fixed size of outgoing packets during the extended transmission (bytes)	500	600	600	600
in_fast_size	The fixed size of incoming packets when outgoing phase is active (bytes)	750	1000	1000	1000
in_slow_size	The fixed size of incoming packets when outgoing phase is inactive (bytes)	250	250	250	250
in_ext_size	The fixed size of incoming packets during the extended transmission (bytes)	600	750	750	750
out_min_len	The minimum length of the outgoing active phase (s)	5	8	8	8
in_delay	The delay of incoming from the start of outgoing. 'null' implies start after outgoing ends (s)	null	2	4	2
initial_max_len	The maximum length of the initial transmission (s)	8	10	15	15
out_interval	The intervals at multiple of which outgoing phase ends (s)	2	2	2	2
ext_interval	The intervals at multiple which extended transmission ends (s)	1	2	2	2

Longer and Varied Interactions. The behavior of skills varies significantly depending on their type. Interactive skills, such as trivia games, may require repeated user interactions, while other skills function more like a simple command, providing a brief response. Additionally, some skills communicate with multiple domains to generate responses (e.g., skills created using VoiceApps, a popular voice app creation platform). This introduces an added layer of complexity when it comes to ensuring robust privacy protections. To allow for this variability, we employ a constant rate bidirectional padded stream as the third phase of our communication. In this setup, real data is sent when available, and dummy data is added when no real data is present. The constant-rate padding ends when the voice assistant completes the command (e.g., drops all underlying connections and enters an 'idle' state). Upon this trigger, padding continues until the next *ext_interval* parameter. This phase of communication is functionally similar to existing constant-rate defenses [12, 13, 21].

Figure 3 provides an overview of PriVA-C in action, demonstrated using two simple commands. The diagram shows how each outgoing phase of communication is padded to the next interval to hide patterns such as the length of the outgoing phase. The subsequent incoming phase is also padded and extended to the next interval to hide any unique sensitive patterns for that particular command. In addition to the aforementioned parameters, we use the rate and size to set the rate of transfer for each state. The full list of parameters along with description can be found in Table 5.

5.2 Evaluation

Similar to our evaluation of existing works in Section 4, we perform a parameter sweep to identify the strongest and lowest-bandwidth variant of PriVA-C, evaluating 1,256 unique combinations. Each platform exhibited distinct traffic characteristics; for example, Alexa generated outgoing packets of approximately 577 bytes every 20 milliseconds. By analyzing these traffic patterns, we fixed certain parameters for each platform (e.g., outgoing phase traffic rate and packet size). For the remaining parameters, given the exponential number of possible combinations, we first determined approximate *operational ranges* (e.g., the outgoing phase typically lasts between 3–5 seconds, we varied it from 1 second to 10 seconds in 1-second increments) by assessing their privacy-utility tradeoffs—varying each parameter individually while keeping others fixed. These tradeoffs were quantified using attack accuracy (Spying attack [4]) and overhead (bandwidth × delay) for each configuration, as shown in Figure 4. The identified ranges were then used in a constrained grid search to determine the final parameter configuration that maximizes the overall privacy-utility tradeoff. We repeat this evaluation for two defense setups: one where the defense is applied only to the primary AVS domain, and another where it is applied to all traffic generated by the voice assistant (see Figure 5).

The analysis reveals that certain parameters—such as traffic rates and packet sizes—tend to increase overhead without offering substantial privacy gains. Intuitively, simply increasing data transmission does not significantly enhance privacy, while reducing it beyond a certain threshold can delay all traffic. This delay may cause shorter voice commands to extend beyond the initial defense window, making them more identifiable to an attacker. At the same time, these parameters must support real-time communication between the voice assistant and its endpoints. To balance privacy and usability, we maintain a traffic rate that matches or slightly exceeds the baseline (undefended) rate, thereby avoiding severe bottlenecks.

We identify a second group of parameters—such as *out_min_len* and *ext_interval*—that tend to improve privacy without significantly increasing overhead. Increasing these parameters reduces the number of distinguishable values observable by an attacker, thereby enhancing privacy. Additionally, we observe that parameters like *in_delay* and *initial_max_len* increase overhead but also contribute to stronger privacy protection. To explore these tradeoffs, we evaluate two defense variants. The first, `Delta`, uses a larger value for *in_delay* to reduce overhead and a larger value for *initial_max_len* to boost privacy. The second, `Forge`, adopts a smaller value for *initial_max_len* to reduce overhead and a smaller value for *in_delay* to enhance privacy. The specific parameter settings for these two variants, along with the strongest variant, named `Strong`, and the lowest bandwidth variant, named `Haven`, are detailed in Table 5.

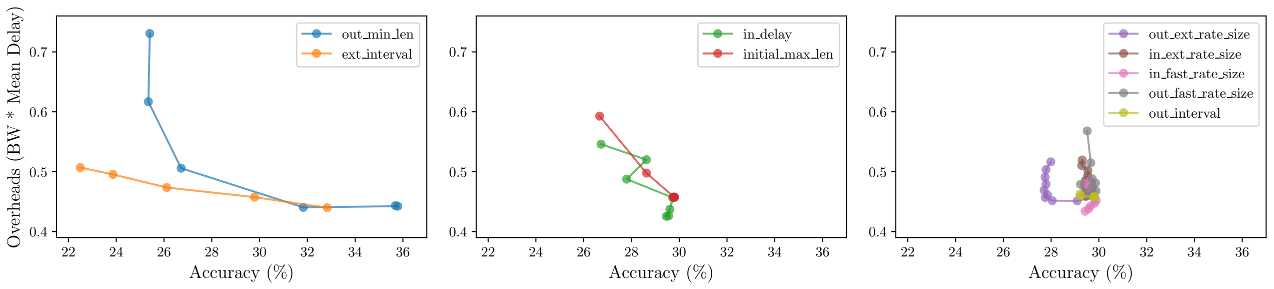
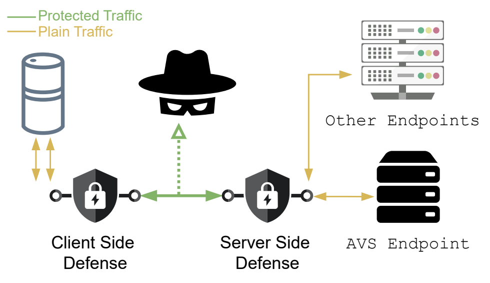
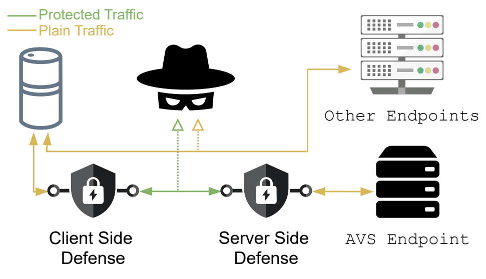


Figure 4: Tradeoff between overheads and attack accuracy when different parameter values are varied.



(a) Defend All Traffic



(b) Defend AVS traffic only

Figure 5: Two different countermeasure setups. (a) shows a setup where all communication between the voice assistant and all internet servers is protected. (b) shows a setup where only the communication between the voice assistant and the primary AVS endpoint is protected using the countermeasure. While setup (a) is expected to be more secure as endpoint information is hidden, it requires the use of a tunneled routed proxy (e.g., VPN or routing all traffic through AVS) which could increase deployment costs for either the user or platform. Setup (b) secures only traffic from the AVS domain and is expected to be slightly less secure but more readily deployable.

Table 6: Results of PriVA-C variants applied to the benchmark dataset protecting "all traffic" from the voice assistant. The results show strong performance with lower overheads.

Setup		Performance			Overheads	
Type	Attack	Accuracy	Precision	Recall	B/W	Delay(s)
Strongest	Spying	12.51%	7.54%	12.91%	156%	3.21
	DeepVC	1.10%	0.01%	1.00%		
Low B/W (Haven)	Spying	22.22%	16.91%	22.50%	125%	3.92
	DeepVC	18.52%	11.31%	18.46%		
`Delta`	Spying	12.51%	7.54%	12.91%	156%	3.21
	DeepVC	1.10%	0.01%	1.00%		
`Forge`	Spying	12.96%	7.32%	13.47%	145%	3.30
	DeepVC	0.90%	0.01%	1.00%		

Table 7: Results of PriVA-C variants applied to the benchmark dataset protecting "only traffic to the AVS domain". The results show strong performance with lower overheads.

Setup		Performance			Overheads	
Type	Attack	Accuracy	Precision	Recall	B/W	Delay(s)
Strongest	Spying	19.42%	19.38%	19.54%	100%	0.73
	DeepVC	10.86%	11.82%	10.71%		
Low B/W (Haven)	Spying	29.88%	26.43%	29.61%	54%	1.40
	DeepVC	24.37%	20.34%	24.59%		
`Delta`	Spying	19.47%	20.05%	19.54%	90%	0.76
	DeepVC	12.16%	11.52%	11.53%		
`Forge`	Spying	19.52%	19.29%	19.69%	76%	0.80
	DeepVC	11.61%	12.26%	11.66%		

The results for the setup that considers all traffic (Table 6) demonstrate that our defense outperforms existing approaches against voice command fingerprinting and achieves comparable effectiveness to constant-rate website fingerprinting defenses, while significantly reducing overhead. Additionally, the results for the setup that defends only traffic to the AVS domain (Table 7) highlight the practicality of our approach in realistic scenarios—where only voice assistant platform providers would need to adjust their communication patterns to enhance user privacy.

To demonstrate the generalizability of our approach, we evaluate PriVA-C on both Google Assistant and Siri platforms. As shown in Table 8, PriVA-C achieves similar performance to Alexa on Siri, reducing fingerprinting accuracy to 20% in its strongest configuration. For Google Assistant, the attack accuracy drops even further to 9%. In both cases, PriVA-C delivers strong privacy protection while significantly lowering overhead compared to existing methods (see Table 3). These results highlight PriVA-C's effectiveness and adaptability across multiple voice assistant platforms.

Overall, the results indicate that our approach offers a promising solution for enhancing user privacy. However, the true impact of the countermeasure—both in terms of privacy and usability—cannot be fully assessed through simulations alone. For instance, a higher overall transmission delay might not affect usability if the content involves large audio files (e.g., a 60-minute podcast), where later chunks can be delayed without interrupting playback due to

Table 8: Results of PriVA-C applied to Google Assistant and Siri protecting "all traffic" from the voice assistants. The results show strong performance with lower overheads compared to existing defenses.

Setup		Google Assistant					Siri				
		Performance			Overheads		Performance			Overheads	
Type	Attack	Accuracy	Precision	Recall	B/W	Delay(s)	Accuracy	Precision	Recall	B/W	Delay(s)
Strongest	Spying	9.22%	4.67%	9.22%	170%	0.35	19.83%	10.46%	19.81%	186%	0.12
	DeepVC	1.20%	0.02%	1.85%	148%	1.36	1.50%	0.03%	2.00%	178%	0.15
Low B/W	Spying	9.40%	5.25%	9.40%	126%	0.82	22.78%	15.52%	22.80%	138%	0.14
	DeepVC	2.10%	0.04%	2.00%	126%	0.82	2.00%	0.06%	2.00%	138%	0.14

Table 9: Results of countermeasure evaluation on real traffic from Alexa prototype modded with PriVA-C. The results show similar performance to simulations.

Setup	Accuracy	Precision	Recall
AVS Domain	25.63%	26.04%	24.84%
All Traffic	17.48%	17.78%	17.00%

pre-loading by the platform. Conversely, even small delays could negatively affect user experience if they cause a noticeable lag in responding to user commands. Parameters like *in_delay* play a critical role in this balance. To validate these assumptions, we develop a real-world prototype (in Section 5.3) and conduct a user study across key defense variants (in Section 5.4).

5.3 Real-World Prototyping

Existing research has highlighted that some defense proposals perform well in simulations but face challenges when deployed on real networks [37, 46]. To evaluate the effectiveness of our proposed defense in a real-world setting, we implemented it on the Alexa platform using the Alexa SDK. The SDK requires deployment on a physical device to enable the installation of MITM proxy certificates and modification of traffic. Since comparable SDKs are not available for Google and Siri, we did not develop prototypes for those platforms. Our implementation is deployed on a Raspberry Pi 4 (4 GB) running the Amazon Alexa SDK [6], to emulate a voice assistant device. Our defense is implemented in Python using a proxy architecture, extending the `mitmproxy` framework [17]. We developed two proxies: a client-side proxy running on the same Raspberry Pi as the client Alexa SDK, and a server-side proxy deployed on a separate Ubuntu server in a different network.

5.4 User Study

We conduct an interactive user study to assess how delays introduced by the proposed countermeasure affect usability.

5.4.1 Recruitment. We recruited 25 participants from the university campus, all over 18 and residing in the U.S. Recruitment was conducted via flyers, university Discord channels, and departmental emails. Eligibility required prior experience with voice assistants, either as standalone devices or integrated into other devices (e.g., mobile phones). Participants provided informed consent, agreed to attend the study in person, and scheduled a timeslot. Of the 29 initial sign-ups, two participants were unable to find a suitable timeslot, and two did not attend. Participant demographics are summarized in Table 11 in Appendix A. The final sample (N = 25) was predominantly Asian (76%), with smaller representation from Caucasian (12%), Black or African American (8%), and Hispanic/Latinx (4%) participants. Most participants were aged 25–34 (72%), with the remainder aged 18–24 (28%). Gender distribution was skewed toward men (64%) compared to women (36%). The sample was highly educated, with 52% holding a bachelor's degree, 44% a master's degree, and 4% a doctorate. While our sample is biased toward younger adults and the Asian population, it included diverse languages and accents, supporting the representativeness and validity of our findings for evaluating PriVA-C.

5.4.2 Study design. To assess the impact of our defense on user experience, we presented participants with four protocol variants. Three variants implemented our countermeasure with different parameter settings (as detailed in Section 5.2) protecting the traffic to AVS domain, while the fourth was unmodified (Alexa's default), serving as a baseline. The baseline was labeled `Crest`, and the defense variants were named `Haven`, `Delta`, and `Forge` (as named in Section 5.2). Participants were unaware of each variant's specific properties and interacted with them in randomized order. Non-sequential pseudonyms were used to prevent users from inferring any delay patterns. Participants were told only that they would test different network protocols affecting device communication. Each participant interacted with all four variants in turn, issuing a fixed set of three predefined commands—"What time is it?", "What is the weather outside?", and "Play Jeopardy". We opted for using three commands per variant to balance time and participant fatigue with reliable results. The three commands selected also have different response lengths in increasing order. Additionally, maintaining the same commands across variants also allows participants to compare variants directly. After each interaction, they rated their experience on a 5-point Likert scale with the statement: "I did not perceive any delay in this interaction." This allowed us to quantify perceived delay across all variants.

After interacting with all four variants in randomized order, participants were informed about how these protocols aim to enhance user privacy. They were then asked which variants they would or would not use, along with their reasoning in *free-text form.* To assess their privacy awareness, participants completed a 7-point IUIPC questionnaire [35]. The study lasted approximately 20-30 minutes per participant, with each receiving a $10 Amazon e-gift card. The study was reviewed and approved by our university's IRB office.

Table 10: Dunn's test post-hoc analysis. Only significant pairs after Bonferroni correction are presented. $^{**}p < 0.01$

Variant 1	Variant 2	p-value
Crest	Haven	**0.0016****
Forge	Haven	**0.0031****

5.4.3 Results. We now compare the ratings across different variants and assess whether the differences are statistically significant.

Ratings. To summarize the variant ratings, we assigned numerical scores from 1 to 5 based on participants' responses on a 5-point Likert scale, with 5 representing the highest rating. Forge and Crest received the highest average rating of 4.24, followed by Delta with 3.88, and Haven with 2.84. These results suggest that participants perceived Forge—the variant with our defense implemented—as the fastest, on par with the default Alexa protocol (Crest). This indicates that our defense did not introduce noticeable delays, preserving usability. In contrast, participants detected delays in Delta and Haven, resulting in lower ratings for those variants.

Statistical Analysis. We conducted statistical analysis to determine whether any variants were significantly more or less preferred by participants. Given the ordinal nature of the 5-point Likert scale ratings, we used the Kruskal-Wallis test, which is appropriate for nonparametric data [23]. Upon finding a statistically significant result, we performed post-hoc analysis using Dunn's test with Bonferroni correction [47]. A *p-value* below 0.05 was considered significant. The Kruskal-Wallis test revealed a statistically significant difference among the variants ($H = 17.0$, $p = 0.0007$). Post-hoc analysis showed that Haven was rated significantly lower than Crest and Forge, while no significant differences were found between Forge, and Delta. This indicates that Haven — the lowest-bandwidth variant — was consistently perceived as slower, while Forge was rated similarly to Alexa's default (Crest) and Delta, confirming that Forge introduced no noticeable lag. The corrected *p-values* for significant comparisons are presented in Table 10.

Unacceptable Variant. When optionally asked which variants they would not use, a significant number of participants (12) indicated they would use all variants. Four participants found Haven unacceptable, three considered Crest (the control variant) unacceptable, and only one participant found Delta unacceptable. No participant found Forge unacceptable. Participants were also asked to provide their rationale. One who chose "Would use all" stated –

"I would choose any options given delay wasn't that much." (P8)

Other opinions include participants who did not like particular variants such as:

"Haven *and* Crest *seemed particularly slow, and I want voice assistants to be as fast as possible with minimal delay. Any sort of perceivable delay makes the process of asking questions less conversational and more cumbersome." (P17)*

Another participant stated that they would not use Haven and identified the artificial delay and stated:

"Haven - *it has a clear 5 second lag" (P18)*

Another participant also singled out Haven, stating

"The response is slow when compared to other variants" (P20)

Overall, most participants identified the variants with clear artificial delays, but a significant number did not find any variant unusable. While some variants, like Haven, were less preferred, other variants were considered acceptable options.

Variant Comparison. Participants responded to the open question about what they think about different variants compared to each other. Participants shared their thoughts about different variants in comparison.

"I perceived more delay in Haven *as compared to the other variants.* Delta *had some delay but it was still a usable system.* Forge *and* Crest *didn't have any delay.* Forge *responses felt instant but the difference between* Forge *and* Crest *was minimal." (P2)*

One participant noted that while some variants were slower than others, the delay was not disruptive. They also compared these variants to another voice assistant they own:

"Crest *and* Forge *were the fastest,* Delta *and* Haven *were slower. I wouldn't say any of them were slower to the point of disruption. My Google assistant takes longer than any of these." (P3)*

One participant specifically mentioned that, despite noticing slight delays in some variants, they would still accept them if they offered better data privacy.

"Forge*, and* Crest *were fast.* Delta *was little slower than the first two but not noticeable.* Haven *was little slow but if it provides me with data privacy I wouldn't mind it giving slow response" (P13)*

IUIPC Score. We presented participants with a 7-point IUIPC questionnaire to assess their perceptions of fairness and privacy across three categories: 1) Control, 2) Awareness, and 3) Collection. The full text of the questions and their average scores are shown in Table 12 in Appendix A. Overall, the IUIPC scores were 6.35 for Control, 6.49 for Awareness, and 5.72 for Collection. These results indicate that participants highly value control over their data, being informed about data handling, and are concerned about excessive data collection, thus, resembling a privacy-conscious mindset.

Takeaway. PriVA-C significantly reduces attacker accuracy while maintaining low overhead. Through a prototype implementation that applies PriVA-C only to the AVS domain, we demonstrate that the defense can be deployed without any noticeable impact on usability. Because our approach requires changes only at the application software level, it enhances deployability—enabling privacy improvements on millions of existing devices through a software update, without the need for modifications to the network stack.

6 Discussion

While previous works have provided privacy guardrails for voice assistants, our approach introduces a strictly improved privacy-preserving defense mechanism that strikes a practical balance between privacy and user experience. Our padding-to-interval approach offers excellent scalability as the number of voice commands and command groups expands. The increasing complexity introduced by a larger command set further confounds attackers, enhancing the overall effectiveness of our defense.

Furthermore, advancements in technology may allow some computational tasks—such as speech-to-text processing—to be offloaded to the device itself, thanks to increasingly powerful on-device machine learning models. While this shift could reduce network overhead, it is unlikely to significantly improve user privacy, as devices will still need to contact network endpoints to retrieve real-time information such as weather, traffic, or news. Moreover, the rise of LLM-powered services [51] may drive more computation back to network servers, as these models remain challenging to run on power-constrained IoT devices.

Recommendations for Deployment. Deploying countermeasures across all network traffic via a proxy, or routing all traffic through voice assistant platform servers, theoretically maximizes endpoint information obfuscation. While adopting this setup is more privacy-preserving, if platform owners route all traffic through their servers, it could increase costs for services [39]. Users of voice assistants can route all their traffic through a VPN proxy and apply this countermeasure even if the platform owners opt not to adopt privacy-enhancing technologies; however, while adept users might be successful in deploying it, an average user might not have the technical prowess to achieve this.

For the aforementioned reasons, we recommend our AVS-only defense, as it provides a slightly less secure, yet significantly more deployable, countermeasure scenario with low overheads and acceptable usability as shown in Section 5.4. This targeted strategy addresses the most critical vulnerabilities without necessitating network reconfiguration, which is a significant advantage over methods that require extensive infrastructure changes. We acknowledge that even with 60-70% bandwidth overhead, it might hinder the adoption cost of operations for platform owners. However, either of these approaches can be provided as a 'premium' privacy feature using a subscription-tier model, as Alexa recently announced with its 'Alexa+', which offers some privacy features [51].

Comparing Alexa SDK vs. Physical Device Traffic. For Alexa, our analysis focuses on traffic generated by the Alexa SDK, which is made available to third-party developers to integrate Alexa functionality into their products. All Alexa enabled devices utilize Amazon Voice Service (AVS), a cloud-based platform that processes user requests [1]. While the traffic exhibits similar structural patterns and phases we conducted a comparative traffic analysis to examine differences between traffic generated by the SDK and by the Amazon Echo device. Figure 6 (a) in Appendix A illustrates domain-level incoming and outgoing traffic differences across selected commands. The Echo device contacts more domains, on average, than the SDK when fulfilling a request. Nevertheless, the overall traffic volume across most commands remains similar between the two platforms (see Figures 6 (b) and Figures 6 (c)). Observed discrepancies are largely attributable to time-sensitive commands, consistent with prior findings by Wang et al. [59], which highlight that such commands evolve over time and may reflect platform-wide changes rather than intrinsic differences between SDK and device traffic. The primary AVS domain also varies between SDK and Echo, a phenomenon also documented in other Alexa-enabled devices such as smartphones [4]. For example, the command "set a timer for 2 minutes" contacts the domain `avs-alexa-4-na.amazon.com` and `unagi-na.amazon.com` on the Echo device. The same commands contacts `avs-alexa-12-na.amazon.com` on the mobile phone Alexa app and `alexa.na.gateway.devices.a2z.com` on the SDK. Despite these variations, traffic from the Echo device remains similarly fingerprintable to that generated by the SDK. In our evaluation of 50 commands, fingerprinting accuracy reached 83.54% for SDK traffic and 79.70% for Echo device traffic. Importantly, our proposed system, PriVA-C, effectively mitigates these risks, reducing fingerprinting accuracy to 10.13% and 8.38%, respectively, for SDK and Echo traffic. In summary, although command traffic may change over time due to evolving responses or platform updates, such changes do not necessitate fundamental alterations in countermeasure design. Instead, they highlight the importance of minor parameter tuning to maintain performance and balance system overhead.

Limitations and Future Work. While our research advances defenses against voice command fingerprinting attacks, several limitations remain. Firstly, the optimization of our defense mechanism, particularly for platforms like Google Assistant and Siri, presents an area for further development. Moreover, dynamically adjusting parameters based on user speech patterns, such as speed and accent, could enhance personalization while reducing overheads. Second, exploring anonymity set techniques, like Super-sequences[41, 60], could strengthen privacy, but they require large databases, which was a practical limitation in our study. Future research should integrate these methods while balancing performance and overhead. Lastly, a few observed corner cases in specific voice assistant third-party skills like waiting for a complete large audio file to load before starting playback might make a fixed rate defense difficult to optimize, since to avoid delays large overheads would be required. However, such cases can be resolved by the platform owner by enforcing playback as each chunk arrives. Despite these limitations, our work lays the groundwork for future research to improve the privacy of voice-controlled systems.

7 Conclusion

In conclusion, this research tackled the critical issue of voice command fingerprinting attacks by thoroughly evaluating existing defense mechanisms. We demonstrated the inadequacy of current voice assistant defenses and explored the applicability of website fingerprinting defenses, revealing their potential yet highlighting inefficiencies in the voice command context.

To address this gap, we introduced a defense mechanism specifically optimized for voice assistants, achieving a strong balance between overheads, privacy, and real-world deployability. We implemented and deployed our solution within an Alexa SDK in a controlled lab environment and conducted a user study to assess the impact on user experience. Our findings emphasize the vulnerability of voice assistants to fingerprinting attacks and the need for specialized defenses. This research proves that an effective and usable defense is possible, underscoring the importance of ongoing research to safeguard the privacy of voice-controlled systems in the future.

Acknowledgments

We thank our anonymous reviewers for their valuable feedback. This research is partially supported by the National Science Foundation (NSF) under grants CNS-2219866 and CNS-2350075. The opinions, findings, conclusions, or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the funding organization.

References

[1] 2025. *Alexa Voice Service v20160207 Alexa Voice Service.* https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/api-overview.html

[2] Humberto J. Abdelnur, Radu State, and Olivier Festor. 2008. Advanced Network Fingerprinting. In *Recent Advances in Intrusion Detection* (Berlin, Heidelberg, 2008), Richard Lippmann, Engin Kirda, and Ari Trachtenberg (Eds.). Springer, 372–389. doi:10.1007/978-3-540-87403-4_20

[3] Dilawer Ahmed, Anupam Das, and Fareed Zaffar. 2022. Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices. 2022, 2 (2022). doi:10.2478/popets-2022-0057

[4] Dilawer Ahmed, Aafaq Sabir, and Anupam Das. 2023. Spying through Your Voice Assistants: Realistic Voice Command Fingerprinting. 2419–2436. https://www.usenix.org/conference/usenixsecurity23/presentation/ahmed-dilawer

[5] Khaled Al-Naami, Amir El-Ghamry, Md Shihabul Islam, Latifur Khan, Bhavani Thuraisingham, Kevin W. Hamlen, Mohammed Alrahmawy, and Magdi Z. Rashad. 2021. BiMorphing: A Bi-Directional Bursting Defense against Website Fingerprinting Attacks. 18, 2 (2021), 505–517. doi:10.1109/TDSC.2019.2907240

[6] alexa. 2025. *avs-device-sdk.* https://github.com/alexa/avs-device-sdk original-date: 2017-02-09T18:57:26Z.

[7] Amazon. 2025. *AI-native SDKs for Alexa+.* https://developer.amazon.com/en-US/alexa/alexa-ai.html

[8] Blake Anderson and David McGrew. 2017. OS fingerprinting: New techniques and a study of information gain and obfuscation. In *2017 IEEE Conference on Communications and Network Security (CNS)* (2017). IEEE, 1–9.

[9] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. (2017). http://arxiv.org/abs/1708.05044

[10] AutoML.org. [n. d.]. *AutoML.* https://www.automl.org/automl/

[11] Charyyev Batyr and Mehmet Hadi Gunes. 2020. Voice Command Fingerprinting with Locality Sensitive Hashes. In *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy* (New York, NY, USA, 2020-11) *(CPSIOTSEC'20)*. Association for Computing Machinery, 87–92. doi:10.1145/3411498.3419963

[12] Xiang Cai, Rishab Nithyanand, and Rob Johnson. 2014. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (Scottsdale Arizona USA, 2014-11). ACM, 121–130. doi:10.1145/2665943.2665949

[13] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. 2014. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale Arizona USA, 2014-11-03). ACM, 227–238. doi:10.1145/2660267.2660362

[14] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. 2012. Touching from a distance: website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and communications security* (New York, NY, USA, 2012-10-16) *(CCS '12)*. Association for Computing Machinery, 605–616. doi:10.1145/2382196.2382260

[15] Heyning Cheng and Ron Avnur. 1998. Traffic Analysis of SSL Encrypted Web Browsing. (1998).

[16] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. 2016. Is Anybody Home? Inferring Activity From Smart Home Network Traffic. In *2016 IEEE Security and Privacy Workshops (SPW)* (2016-05). 245–251. doi:10.1109/SPW.2016.48

[17] Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors. 2010. *mitmproxy - an interactive HTTPS proxy.* https://mitmproxy.org/

[18] Shuaifu Dai, Alok Tongaonkar, Xiaoyin Wang, Antonio Nucci, and Dawn Song. 2013. NetworkProfiler: Towards automatic fingerprinting of Android apps. In *2013 Proceedings IEEE INFOCOM* (2013-04). 809–817. doi:10.1109/INFCOM.2013.6566868 ISSN: 0743-166X.

[19] Wladimir De la Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. 2020. TrafficSliver: Fighting Website Fingerprinting Attacks with Traffic Splitting. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 1971–1985. doi:10.1145/3372297.3423351

[20] Daniel J. Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. 2020. When Speakers Are All Ears: Characterizing Misactivations of IoT Smart Speakers. 2020, 4 (2020). doi:10.2478/popets-2020-0072

[21] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. 2012. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *2012 IEEE Symposium on Security and Privacy* (2012-05). 332–346. doi:10.1109/SP.2012.28 ISSN: 2375-1207.

[22] Anna Fleck. 2024. *Infographic: Alexa, What's America's Favorite Smart Speaker?* https://www.statista.com/chart/23943/share-of-us-adults-who-own-smart-speakers

[23] Jim Frost. 2017. *Nonparametric Tests vs. Parametric Tests.* http://statisticsbyjim.com/hypothesis-testing/nonparametric-parametric-tests/

[24] Xiaoguang Guo, Keyang Yu, Qi Li, and Dong Chen. 2024. VoiceAttack: Fingerprinting Voice Command on VPN-protected Smart Home Speakers. In *Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (New York, NY, USA, 2024-10-29) *(BuildSys '24)*. Association for Computing Machinery, 55–65. doi:10.1145/3671127.3698171

[25] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. 1187–1203. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/hayes

[26] Sébastien Henri, Ginés García, Pablo Serrano, Albert Banchs, and Patrick Thiran. 2020. Protecting against website fingerprinting with multihoming. 2020, 2 (2020), 89–110. doi:10.2478/popets-2020-0019

[27] Spherical Insights. [n. d.]. *Global Voice Assistant Market Size, Share, Forecast to 2033.* https://www.sphericalinsights.com/reports/voice-assistant-market

[28] Umar Iqbal, Pouneh Nikkhah Bahrami, Rahmadi Trimananda, Hao Cui, Alexander Gamero-Garrido, Daniel Dubois, David Choffnes, Athina Markopoulou, Franziska Roesner, and Zubair Shafiq. 2022. Your Echos are Heard: Tracking, Profiling, and Ad Targeting in the Amazon Smart Speaker Ecosystem. (2022). arXiv:2204.10920 http://arxiv.org/abs/2204.10920

[29] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. 2016. Toward an efficient website fingerprinting defense. In *European Symposium on Research in Computer Security* (2016). Springer, 27–46.

[30] Sean Kennedy, Haipeng Li, Chenggang Wang, Hao Liu, Boyang Wang, and Wenhai Sun. 2019. I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers. In *2019 IEEE Conference on Communications and Network Security (CNS)* (2019-06). 232–240. doi:10.1109/CNS.2019.8802686

[31] Hyojin Kim, Minji Jo, Jiwoo Hong, Hosung Kang, Nate Mathews, and Se Eun Oh. 2024. WhisperVoiceTrace: A Comprehensive Analysis of Voice Command Fingerprinting. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security* (New York, NY, USA, 2024-07-01) *(ASIA CCS '24)*. Association for Computing Machinery, 667–683. doi:10.1145/3634737.3657017

[32] Christopher Lentzsch, Sheel Jayesh Shah, Benjamin Andow, Martin Degeling, Anupam Das, and William Enck. 2021. Hey Alexa, is this skill safe?: Taking a closer look at the Alexa skill ecosystem. In *28th Annual Network and Distributed System Security Symposium, NDSS* (2021).

[33] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. 5 (2017), 18042–18050. doi:10.1109/ACCESS.2017.2747560 Conference Name: IEEE Access.

[34] David Lu, Sanjit Bhat, Albert Kwon, and Srinivas Devadas. 2018. DynaFlow: An Efficient Website Fingerprinting Defense Based on Dynamically-Adjusting Flows. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society* (New York, NY, USA, 2018-01-15) *(WPES'18)*. Association for Computing Machinery, 109–113. doi:10.1145/3267323.3268960

[35] Naresh K. Malhotra, Sung S. Kim, and James Agarwal. 2004. Internet Users' Information Privacy Concerns (IUIPC): The Construct, the Scale, and a Causal Model. 15, 4 (2004), 336–355. doi:10.1287/isre.1040.0032 Publisher: INFORMS.

[36] Jianghan Mao, Chenyu Wang, Yanhui Guo, Guoai Xu, Shoufeng Cao, Xuanwen Zhang, and Zixiang Bi. 2022. A novel model for voice command fingerprinting using deep learning. 65 (2022), 103085. doi:10.1016/j.jisa.2021.103085

[37] Nate Mathews, James K Holland, Se Eun Oh, Mohammad Saidur Rahman, Nicholas Hopper, and Matthew Wright. [n. d.]. SoK: A Critical Evaluation of Efficient Website Fingerprinting Defenses. ([n. d.]).

[38] Takashi Matsunaka, Akira Yamada, and Ayumu Kubota. 2013. Passive OS fingerprinting by DNS traffic analysis. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)* (2013). IEEE, 243–250.

[39] Dana Mattioli. 2024. *Alexa Is in Millions of Households—and Amazon Is Losing Billions.* https://www.wsj.com/tech/amazon-alexa-devices-echo-losses-strategy-25f2581a Section: Tech.

[40] Milad Nasr, Alireza Bahramali, and Amir Houmansadr. 2021. Defeating DNN-Based Traffic Analysis Systems in Real-Time With Blind Adversarial Perturbations. In *30th USENIX Security Symposium (USENIX Security 21)* (2021). 2705–2722. https://www.usenix.org/conference/usenixsecurity21/presentation/nasr

[41] Rishab Nithyanand, Xiang Cai, and Rob Johnson. 2014. Glove: A Bespoke Website Fingerprinting Defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society* (Scottsdale Arizona USA, 2014-11-03). ACM, 131–134. doi:10.1145/2665943.2665950

[42] nmap.org. 1997. *Nmap: the Network Mapper - Free Security Scanner.* https://nmap.org/

[43] Tajammul Pangarkar. 2025. *Smart Speaker Statistics and Facts (2025).* https://scoop.market.us/smart-speaker-statistics/

[44] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. 2020. IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)* (2020-09). 474–489. doi:10.1109/EuroSP48549.2020.00037

[45] Mohammad Saidur Rahman, Mohsen Imani, Nate Mathews, and Matthew Wright. 2021. Mockingbird: Defending Against Deep-Learning-Based Website Fingerprinting Attacks With Adversarial Traces. 16 (2021), 1594–1609.

doi:10.1109/TIFS.2020.3039691 Conference Name: IEEE Transactions on Information Forensics and Security.

[46] Mohammad Saidur Rahman, Payap Sirinam, Nate Mathews, Kantha Girish Gangadhara, and Matthew Wright. 2020. *Tik-Tok* : The Utility of Packet Timing in Website Fingerprinting Attacks. 2020, 3 (2020), 5–24. doi:10.2478/popets-2020-0043

[47] Ash Rajesh. 2023. *A Post-hoc Test for Kruskal-Wallis.* https://www.theanalysisfactor.com/dunns-test-post-hoc-test-after-kruskal-wallis/

[48] Aafaq Sabir, Evan Lafontaine, and Anupam Das. 2022. Hey Alexa, Who Am I Talking to?: Analyzing Users' Perception and Awareness Regarding Third-party Alexa Skills. In *CHI Conference on Human Factors in Computing Systems* (2022).

[49] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J. Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. 2020. A Haystack Full of Needles: Scalable Detection of IoT Devices in the Wild. In *Proceedings of the ACM Internet Measurement Conference* (New York, NY, USA, 2020-10) *(IMC '20)*. Association for Computing Machinery, 87–100. doi:10.1145/3419394.3423650

[50] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. 2007. Devices That Tell on You: Privacy Trends in Consumer Ubiquitous Computing. https://www.usenix.org/conference/16th-usenix-security-symposium/devices-tell-you-privacy-trends-consumer-ubiquitous

[51] Panos Panay Services, SVP of Devices \&. 2025. *Introducing Alexa+, the next generation of Alexa.* https://www.aboutamazon.com/news/devices/new-alexa-generative-artificial-intelligence

[52] Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov. 2014. Hershel: single-packet os fingerprinting. 42, 1 (2014), 195–206. Publisher: ACM New York, NY, USA.

[53] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: attacks and defenses. In *Proceedings of the 11th European conference on Research in Computer Security* (Berlin, Heidelberg, 2006-09-18) *(ESORICS'06)*. Springer-Verlag, 18–33.

[54] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018-10-15) *(CCS '18)*. Association for Computing Machinery, 1928–1943. doi:10.1145/3243734.3243768

[55] Payap Sirinam, Nate Mathews, Mohammad Saidur Rahman, and Matthew Wright. 2019. Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with N-shot Learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2019-11-06) *(CCS '19)*. Association for Computing Machinery, 1131–1148. doi:10.1145/3319535.3354217

[56] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-Level Signatures for Smart Home Devices. In *Proceedings 2020 Network and Distributed System Security Symposium* (San Diego, CA, 2020). Internet Society. doi:10.14722/ndss.2020.24097

[57] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Mérindol, and Benoit Donnet. 2013. Network fingerprinting: TTL-based router signatures. In *Proceedings of the 2013 conference on Internet measurement conference* (New York, NY, USA, 2013-10-23) *(IMC '13)*. Association for Computing Machinery, 369–376. doi:10.1145/2504730.2504761

[58] Wang. 2024. *SmartHomePrivacyProject/DeepVCFingerprinting.* https://github.com/SmartHomePrivacyProject/DeepVCFingerprinting original-date: 2020-05-06T17:47:22Z.

[59] Chenggang Wang, Sean Kennedy, Haipeng Li, King Hudson, Gowtham Atluri, Xuetao Wei, Wenhai Sun, and Boyang Wang. 2020. Fingerprinting encrypted voice traffic on smart speakers with deep learning. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (New York, NY, USA, 2020-07) *(WiSec '20)*. Association for Computing Machinery, 254–265. doi:10.1145/3395351.3399357

[60] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. 143–157. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao

[61] Tao Wang and Ian Goldberg. 2017. {Walkie-Talkie}: An Efficient Defense Against Passive Website Fingerprinting Attacks. 1375–1390. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

A Appendix

Table 11: Demographics of participants.

Demographic	Value	Count	Percentage
Age Group	25-34	18	72%
	18-24	7	28%
Gender	Man	16	64%
	Woman	9	36%
Education	Bachelor's degree	13	52%
	Master's degree	11	44%
	Doctorate degree	1	4%
Race	Asian	19	76%
	Caucasian	3	12%
	Black or African American	2	8%
	Hispanic, Latinx, or Spanish origin	1	4%

Table 12: IUIPC scores across three dimensions: Control, Awareness, and Collection.

Dimension	Question Summary	Mean Score	Section Mean
Control	Q1: Right to control information	6.35	6.35
	Q2: Control central to privacy	6.39	
	Q3: Loss of control invades privacy	6.30	
Awareness	Q4: Data disclosure practices	6.78	6.49
	Q5: Clear privacy policies	6.87	
	Q6: Importance of knowledge about data use	5.83	
Collection	Q7: Discomfort with sharing info	5.43	5.72
	Q8: Hesitation to provide info	5.74	
	Q9: Concern over giving info to many companies	6.0	
	Q10: Concern over excessive data collection	5.7	

Table 13: Results of using different end classification models for 'Spying' attack with PriVA-C protecting "all traffic" from the voice assistant. The results show similar performance with very slight improvement for AutoGluon

Variant	Model	Accuracy	Precision	Recall
Strongest	Random Forest	12.51%	7.54%	12.91%
	AutoGluon	12.61%	7.53%	13.00%
Low B/W (Haven)	Random Forest	22.22%	16.91%	22.50%
	AutoGluon	22.92%	16.57%	22.81%
Delta	Random Forest	12.51%	7.54%	12.91%
	AutoGluon	12.61%	7.53%	13.00%
Forge	Random Forest	12.96%	7.32%	13.47%
	AutoGluon	13.08%	8.17%	13.54%

Table 14: Results of using different end classification models for 'Spying' attack with PriVA-C protecting "only traffic to primary AVS domain". The results show similar performance with very slight improvement for AutoGluon

Variant	Model	Accuracy	Precision	Recall
Strongest	Random Forest	19.42%	19.38%	19.54%
	AutoGluon	19.77%	20.26%	19.88%
Low B/W (Haven)	Random Forest	29.88%	26.43%	29.61%
	AutoGluon	30.22%	26.99%	30.02%
Delta	Random Forest	19.76%	20.05%	19.54%
	AutoGluon	21.10%	20.85%	20.43%
Forge	Random Forest	19.52%	19.29%	19.69%
	AutoGluon	20.77%	20.49%	20.39%

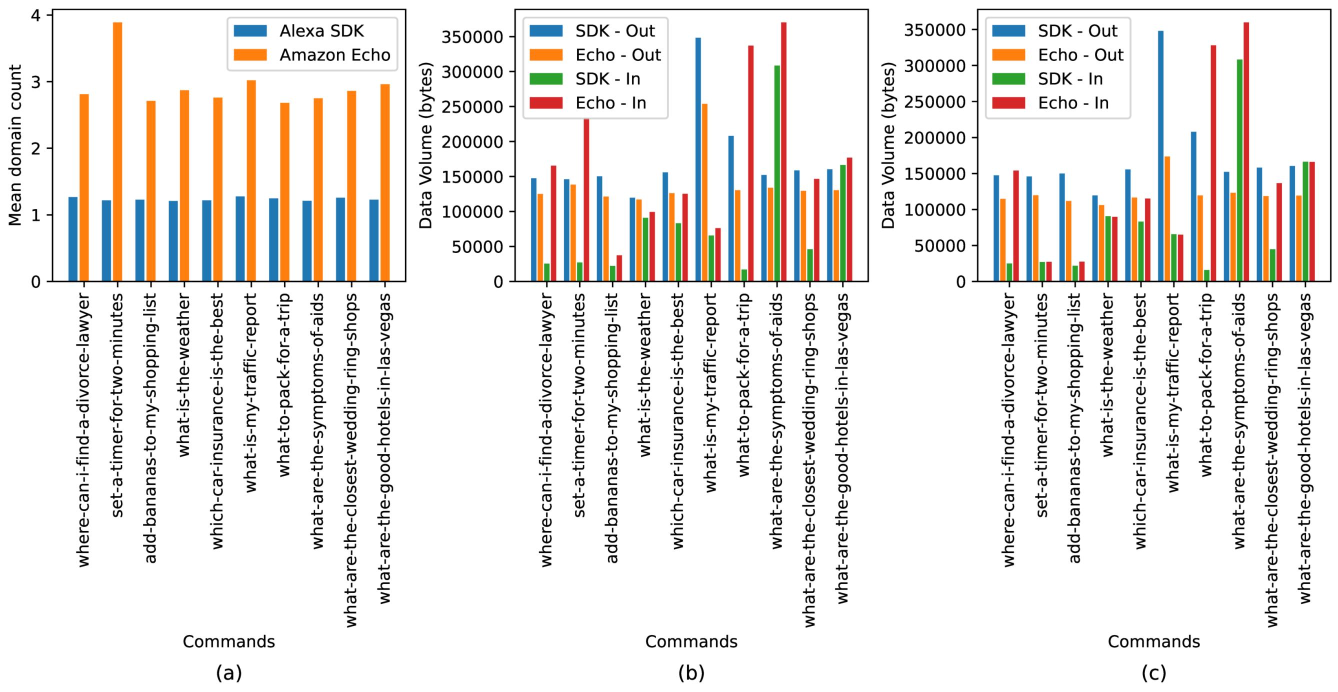


Figure 6: Comparison of traffic from Alexa SDK and Amazon Echo device. (a) shows that traffic from Amazon Echo on average contacts more domains compared to traffic from SDK. (b) shows traffic volume in outgoing and incoming directions from SDK and Amazon Echo to all domains. For most commands traffic patters are similar whereas for remaining we see some variance in traffic. (c) shows traffic volume to primary domain only.

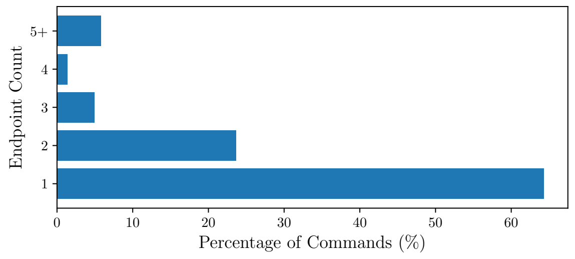


Figure 7: 65% of commands contact only one AVS Endpoint.