Towards Privacy-Preserving and Personalized Smart Homes via Tailored Small Language Models

Xinyu Huang, *Graduate Student Member, IEEE,* Leming Shen, *Graduate Student Member, IEEE,* Zijing Ma, *Graduate Student Member, IEEE,* Yuanqing Zheng, *Senior Member, IEEE,*

***Abstract*—Large Language Models (LLMs) have showcased remarkable generalizability in language comprehension and hold significant potential to revolutionize human-computer interaction in smart homes. Existing LLM-based smart home assistants typically transmit user commands, along with user profiles and home configurations, to remote servers to obtain personalized services. However, users are increasingly concerned about the potential privacy leaks to the remote servers. To address this issue, we develop *HomeLLaMA*, an on-device assistant for privacy-preserving and personalized smart home serving with a tailored small language model (SLM). *HomeLLaMA* learns from cloud LLMs to deliver satisfactory responses and enable user-friendly interactions. Once deployed, *HomeLLaMA* facilitates proactive interactions by continuously updating local SLMs and user profiles. To further enhance user interaction while protecting their privacy, we develop *PrivShield* to offer an optional, privacy-preserving LLM-based smart home service for users who are unsatisfied with local responses and are willing to send less-sensitive queries to remote servers. For evaluation, we develop a comprehensive benchmark, *DevFinder*, to assess service quality. Extensive experiments and user studies ($M = 100$) demonstrate that *HomeLLaMA* can provide personalized services while significantly enhancing user privacy.**

***Index Terms*—Smart Home, Large Language Model, Privacy, Personalization**

I. Introduction

THE proliferation of smart homes has significantly facilitated the development of intelligent living spaces [1], [2]. Typically, a smart home is a residence equipped with various interconnected devices and systems [3], [1], [4] that can be controlled remotely or autonomously to enhance efficiency and convenience through technologies such as IoT and AI-based chatbots [5], [6]. The long-term goal of smart homes is to achieve seamless user-assistant interaction, allowing systems to deeply comprehend user intents and deliver satisfactory and personalized responses [7].

Existing commercial-off-the-shelf (COTS) smart home assistants, like Amazon Alexa [8] and Apple Siri [9], are task-specific models pretrained on various instruction datasets, which may lead to degraded performance on unseen tasks. For instance, when users provide an under-specified command (*e.g.*, "Let guests in") without mentioning specific devices, the system might struggle to generate a reasonable action plan involving smart devices due to the lack of a predefined command. Consequently, users and developers must add new command-action pairs manually to customize the assistant. The configuration process [10] mainly involves setting up triggers (*e.g.*, specific times and conditions) and defining corresponding actions (*e.g.*, starting appliances and predefined routines), which are complex and time-consuming for application developers, let alone novice users.

To overcome these limitations, recent works integrate LLMs [11], [12] to revolutionize smart home services, enabling assistants to understand user intents beyond predefined commands. Among them, Sasha leverages ChatGPT [13] to generate action plans in response to user commands. SAGE further enhances user experience by storing conversation histories for personalized plan generation. Nevertheless, these cloud LLM-based assistants introduce substantial privacy risks. In practice, users typically need to register for API keys to access the cloud LLM services. During operation, user commands, constructed profiles, and home device states (*e.g.*, on/off state of lights) are transmitted to the cloud for processing. Under the *honest-but-curious* threat model [14], [**?**], [15], [10], this workflow may expose user privacy [16], [17], including daily routines (*e.g.*, cooking, exercising), personal preferences, and detailed home configurations, to third parties.

To protect user privacy from being exposed, an alternative approach is to exploit open-source models to serve smart homes locally. Though promising, local devices can only support small-sized language models (SLMs) due to resource constraints. [18], [19], [20] Our preliminary study (§ III-B) reveals that SLMs often fall short in fully comprehending user intents [21]. Therefore, users are facing a *performance-privacy dilemma*: while cloud LLMs excel in delivering high-quality services, they may raise privacy concerns; conversely, local SLMs secure user privacy but fail to generate satisfactory responses due to limited model capabilities.

To address this dilemma, we propose *HomeLLaMA*, a privacy-preserving local home assistant that delivers personalized and satisfactory services through continuous learning. The key insight of *HomeLLaMA* is empowering local SLMs with the capabilities of cloud LLMs to shift most privacy-sensitive query processing tasks from the cloud to the local, thereby achieving a balance between model performance and user privacy. The powerful cloud services are consulted with users' explicit approval only when necessary (*e.g.*, unsatisfactory responses of the local SLMs). While the basic idea is simple, several technical challenges must be addressed.

- *SLMs perform poorly compared to LLMs and lack high-*

This work is supported by Hong Kong GRF under Grant No. 15206123 and 15211924.

X. Huang, L. Shen, Z. Ma, and Y. Zheng are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China. E-mail: unixy-xinyu.huang@connect.polyu.hk, leming.shen@connect.polyu.hk, zijing.ma@connect.polyu.hk, and csyqzheng@comp.polyu.edu.hk.

Corresponding author: Yuanqing Zheng.

quality datasets for effective enhancement.

Preliminary experiments (§ III-B) reveal that the key bottleneck of SLMs in delivering high-quality smart home plans lies in their limited capabilities to accurately associate relevant devices with user commands compared with cloud LLMs. Yet further experiments show that directly tuning SLMs on existing command-action pairs only yields slight performance gains due to inadequate generalizability across heterogeneous home configurations. To address it, we propose a novel labor-free data augmentation method with a tailored inference paradigm. Specifically, we instruct powerful cloud LLMs to synthesize a generalizable command-device dataset based on available crowdsourced data. We fine-tune local SLMs on such a synthesized dataset and guide them using the consistent inference pipeline with well-crafted prompts. As a result, the fine-tuned local SLMs can effectively generate higher-quality action plans that are applicable across diverse homes with varied device configurations.

- *Even a well-enhanced SLM may not consistently provide cloud LLM-level services for users.*

As shown in the preliminary results (Fig. 2(b)), even after fine-tuning with our well-constructed dataset, there still remains a substantial performance gap between the local SLM and the cloud LLM. This gap in serving smart homes may undermine user experience, limited by local SLMs. To resolve this issue, we design a privacy-preserving local-cloud collaboration paradigm, providing users with the option to consult cloud assistance for higher-quality responses. During this collaboration, *HomeLLaMA* retains user preference and home configuration data locally, and further obfuscates the commands sent for assistance to preserve user privacy. The process is entirely user-driven, meaning that the privacy-sensitive commands will only be processed and then transmitted to remote servers for performance enhancement upon explicit user approval.

- *Limited local space for guaranteeing long-term personalized services.*

Prior work [22] embeds entire historical user-assistant conversations into prompts for personalization. However, open-source SLMs have a shorter context length (*e.g.*, 8K tokens for LLaMA3) than commercial models and cannot incorporate long conversation histories into prompts. While the popular retrieval-augmented generation (RAG) [23] is promising in reducing context length by fetching relevant information from a database, merely storing all historical conversations in the database can lead to the continuous accumulation of preference-related data, resulting in redundancy and hindering the efficient retrieval of highly correlated information. To mitigate this, at the end of each conversation, we instruct the local SLM to distill the current chat into a concise user profile containing topics, preferences, the current command, and its final approved plan. Following the digestion of historical data, we design a dynamic profile updating mechanism based on similarity to reduce redundancy.

We implement and deploy *HomeLLaMA* on a local server concerning specific smart home layouts and evaluate its performance across multiple commonly used scenarios (*e.g.*, atmosphere adjustment, and energy management). Both quantitative experiments and sufficient user studies ($M = 100$) reveal *HomeLLaMA* significantly enhances user-centered privacy while maintaining an acceptable level of performance, alleviating the raised performance-privacy dilemma for smart home users. In short, the **contributions** are as follows:

- To the best of our knowledge, *HomeLLaMA*[1] is the first on-device smart home assistant to support privacy-preserving and personalized services via user-in-the-loop.
- *HomeLLaMA*[2] features three novel technical modules: *Local SLM Enhancement* for effectively enhancing the performance of local assistants with a tailored inference paradigm, *Local-Cloud Collaboration* for maximizing user experience via a user-centered local-cloud collaborative workflow with privacy considerations, and *User Preference Learning* for efficient locally-hosted personalization.
- We build a comprehensive benchmark *DevFinder*[3] to quantitatively evaluate the performance of smart assistants. Extensive experiments and user studies demonstrate *HomeLLaMA* offers satisfactory and privacy-enhanced services.

II. Related Work

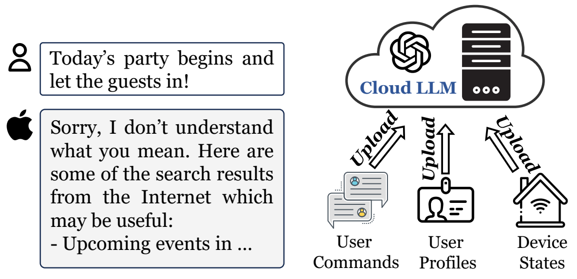
A. Smart Homes

In recent years, smart homes have emerged as a significant area of interest within the broader domain of Internet of Things (IoT) [26]. These systems integrate various connected devices to automate and enhance home living, offering functionalities such as energy management [1] and personalized services [27]. A typical scenario contains several smart devices, a user interface component, and a central processing unit that connects the smart home with cloud servers [28]. On the smart device side, recent research has focused on enhancing device capabilities through machine learning algorithms. For instance, [29] explores activity recognition for home automation by developing a deep learning algorithm that identifies user activities based on accelerometer data collected by devices. On the user interface side, voice-based assistants are increasingly preferred due to their ability to facilitate natural language interactions and hands-free control. Commercial products like Google Assistant [30], and Alexa [8] exemplify this trend, offering intuitive interfaces capable of managing various commands, such as shopping and setting reminders, to streamline automated device control. However, these modern home assistants usually struggle with implicit and complex commands [31], as demonstrated in our preliminary study. To address this issue, traditional methods design tailored machine learning frameworks to learn user preferences and adapt to heterogeneous smart homes [32], [33], [34]. However, they still require specialized fine-tuning or other adjustments when deployed in unseen scenarios with data domain shifts. On the other hand, recent advances in LLMs have shown excellent performance in open-vocabulary question answering, which can better comprehend user intentions with under-specified

[1]The trained model: https://huggingface.co/USER9724/HomeLlama-8B.
[2]The code: https://github.com/unixyhuang/homellama.
[3]https://huggingface.co/datasets/USER9724/SmartHome-Device-QA

TABLE I
A COMPREHENSIVE COMPARISON WITH OTHER LLM-BASED ASSISTANTS.

System	Base Model	Plan Quality	Personalization	Privacy Protection	User Engagement
HomeGPT [24]	GPT-3.5 (Cloud)	Medium	Limited	Low	Limited
Sasha [7]	GPT-4 (Cloud)	High	Limited	Low	Limited
SAGE [22]	GPT-4 (Cloud)	High	High	Low	Limited
TT-Gemma [25]	Gemma (Local)	Low	Limited	High	Limited
TT-Phi-2 [25]	Phi-2 (Local)	Low	Limited	High	Limited
HomeLLaMA	**LLaMA3 (Local)**	**Medium**	**High**	**High ↑**	**Proactive**


Today's party begins and let the guests in!

Sorry, I don't understand what you mean. Here are some of the search results from the Internet which may be useful:
- Upcoming events in ...

(a) A typical example of Siri. (b) General workflow of SOTA.

Fig. 1. Illustrations of existing works.

commands. *HomeLLaMA* enhances user experiences with improved system performance by integrating LLMs with smart home devices to overcome the aforementioned challenges.

B. Integrate LLMs with Smart Homes

Recognizing the strong generalizability and language processing capabilities of LLMs [35], [36], [37], researchers are attempting to integrate them with smart homes for enhanced user experiences. A pioneering work, HomeGPT [24], directly prompts LLMs to generate a series of routines for the smart devices by providing the user command with detailed device states. The routines will further be parsed to adjust the states of the smart devices accordingly. Sasha [7] further optimizes the inference and control procedures by dividing the entire process into five steps: clarifying, filtering, planning, execution, and feedback. Nonetheless, it cannot adapt to user habits to generate personalized action plans, lowering long-term user satisfaction. To address this, SAGE [22] and Jordan *et al.* [38] enable LLMs to incorporate user profiles for generating personalized plans. However, these systems transmit user data and smart home configurations to the cloud LLM for processing, raising privacy concerns for users as the data exits the local environment. To provide satisfactory and personalized plans while enhancing privacy, *HomeLLaMA* tailors a locally deployed SLM via fine-tuning, focusing on providing satisfactory and personalized smart home plans while enhancing user privacy through our designed *PrivShield*.

In summary, Table I qualitatively presents a comprehensive comparison between *HomeLLaMA* and other LLM-based smart home assistants across multiple dimensions. Each of these dimensions corresponds to specific quantitative metrics discussed in the evaluation section (§ VI), for example, *plan quality* is measured using the defined *Device Relevance Score*. Compared with cloud-based assistants, *HomeLLaMA* offers personalized services while significantly enhancing user privacy with minimal performance trade-offs. On the other hand, compared with local-based solutions, *HomeLLaMA* excels in providing superior personalization and higher-quality plans. Additionally, it favors an innovative interaction paradigm that promotes proactive user engagement through a user-driven user-assistant interaction chain, enabling users to actively personalize responses for a more adaptive experience.

C. Privacy-Preserving LLM Services

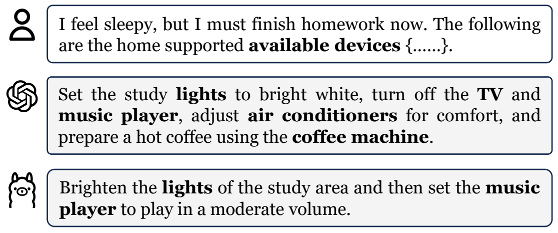
As LLM-based services become increasingly integrated into various workplaces, they also raise significant privacy concerns [39] due to the transmission of sensitive user queries to remote servers for processing. To address these concerns, beyond computationally intensive encryption techniques such as homomorphic encryption [40], several approaches have been proposed to sanitize user prompts. For instance, Cape [41] employs differential privacy to perturb transmitted prompts, thereby protecting users' personal attributes (*e.g.*, gender, age) from inference by cloud-based adversaries; InferGPT [42] replaces identified sensitive information with alternative content to sanitize prompts; and GPTWall [43] utilizes an edge-deployed LLM to enable users to customize their privacy preferences. However, these methods primarily target general-use scenarios and focus on safeguarding user attributes or identity information. In contrast, home-related applications represent a particularly privacy-sensitive domain where general-purpose privacy-preserving techniques may prove inadequate.

In contrast, *HomeLLaMA* introduces a tailored privacy-preserving framework specifically designed for LLM-based smart home services. At its core, *HomeLLaMA* leverages an enhanced local SLM to process and store the majority of user information locally. Additionally, when explicitly authorized by users, it employs an obfuscation-based technique to rewrite and obscure user queries, offering an extra layer of privacy protection that safeguards real-time in-home activities beyond what general privacy-preserving approaches typically provide.

III. MOTIVATION AND CHALLENGES

A. Limitations of Existing Smart Home Assistants

Existing solutions for smart assistants can be categorized into task-based and LLM-based assistants. Task-based assis-

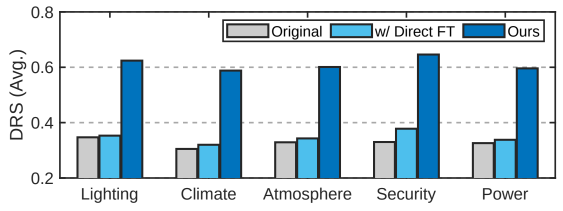


I feel sleepy, but I must finish homework now. The following are the home supported **available devices** {......}.

Set the study **lights** to bright white, turn off the **TV** and **music player**, adjust **air conditioners** for comfort, and prepare a hot coffee using the **coffee machine**.

Brighten the **lights** of the study area and then set the **music player** to play in a moderate volume.

(a) Responses from GPT-4 and LLaMA3 give a command.



(b) Average *DRS* results across multiple scenarios.

Fig. 2. Preliminary results of (a) responses from GPT-4 and LLaMA3 and (b) *DRS* after setting GPT-4 as references.

tants are trained on predefined command-action pairs, while LLM-based assistants utilize the robust capabilities of LLMs to understand user intents in various smart home scenarios.

Limitations of existing task-based assistants. As a notable task-based assistant trained on a vast human-annotated dataset, Siri can deliver excellent responses to predefined tasks [9]. However, its performance degrades when encountering unseen and complex commands. Fig. 1(a) illustrates a typical failure scenario in a conversation between Apple Siri and a smart home user. When the user inputs a command such as ”Party begins and let all the guests in!” Siri fails to provide an appropriate response and instead directly returns the search results from the Internet, leading to a poor user experience.

Privacy concerns of LLM-based assistants. Recent advancements in LLM-based smart home assistants, such as Sasha and SAGE, allow users to issue commands more freely and receive responses that go beyond predefined tasks. Specifically, Sasha prompts the LLM using a designed pipeline with steps like clarifying, filtering, and planning to generate satisfactory action plans in response to user commands. Sasha fails to provide personalized services. On the other hand, SAGE further enhances personalization by storing conversation histories and summarizing them into user profiles.

Despite the improvements, as illustrated in Fig. 1(b), this workflow may pose significant risks to user privacy. In practice, users are required to transmit commands along with constructed user profiles and detailed device states (*e.g.*, a JSON file indicating the status of an air conditioner) to cloud servers for processing. Assuming an *honest-but-curious* cloud adversary [14], this workflow may lead to the exposure of:

- Sensitive personal information, including personally identifiable information (PII) and user preferences [44];
- Home configurations, *e.g.*, real-time home device states [45];
- Users’ daily in-home activities/routines, *e.g.*, exercising [46].

These privacy risks hinder existing LLM-based assistants.

B. Challenges

As a straightforward solution to mitigate privacy concerns of existing LLM-based assistants, we conduct preliminary experiments by deploying the open-source LLaMA3-8B [47] on a local server. From the preliminary study, we report several technical challenges that further inspire *HomeLLaMA*.

Challenge 1: Vanilla SLMs perform poorly in identifying relevant devices and lack high-quality datasets for effective fine-tuning. To first uncover the underlying reasons why SLMs underperform LLMs in smart homes qualitatively, we input an under-specified command along with a set of available devices to both GPT-4 and LLaMA3 to generate responses. As shown in Fig. 2(a), GPT-4 involves a comprehensive list of relevant devices, whereas LLaMA3 generates a simpler response, mentioning only lights and the music player. The result suggests that SLMs mainly lack the inherent capability and domain knowledge in identifying the latent semantic correlation between user commands and relevant devices.

Given this observation, a user-configured dataset from smart home platforms [48] is then collected for fine-tuning the SLM. Once tuned, we input the prepared test commands into both the original and the fine-tuned SLM in the same prompt format, generating two sets of relevant devices as responses. For a fair comparison, the same test commands are also processed using GPT-4 to produce reference device outputs. All responses are generated based on a predefined device set, thereby constraining the models from generating outputs in a freestyle manner. To quantify each model’s ability to associate relevant devices with input commands, we adopt the *device relevance score (DRS)* defined in [7], with a detailed metric definition provided in § VI. As shown in Fig. 2(b), the comparison reveals that *DRS* values only exhibit a slight improvement (less than 10%) across various scenarios after tuning on the dataset. The minimal performance gain from the existing crowd-sourced dataset drives us to construct a high-quality dataset tailored for fine-tuning SLMs in smart homes.

Challenge 2: Even a well-enhanced SLM may not consistently generate cloud LLM-level responses. We explore potential strategies to address **Challenge 1** and finally construct a fit-for-purpose dataset for effectively enhancing SLMs in the context of smart homes (§ IV-B). However, as demonstrated in Fig. 2(b), although fine-tuning SLMs with our tailored dataset significantly improves performance, a gap still remains between local SLMs and cloud LLMs. In practice, this implies that even with enhancement, SLMs may still fail to consistently deliver high-quality services to users in smart home environments. Such inconsistencies can diminish the user experience, particularly when compared to the robust cloud-based assistants. The gap highlights the need for a user-driven cloud-assisted mechanism—one that incorporates privacy-preserving measures—enabling users to obtain higher-quality responses when local outputs fail to meet expectations.

Challenge 3: Simply storing all interaction history for personalization necessitates an excessively long context. Existing approaches [22] directly incorporate raw conversation history or accumulated user profiles into prompts for cloud

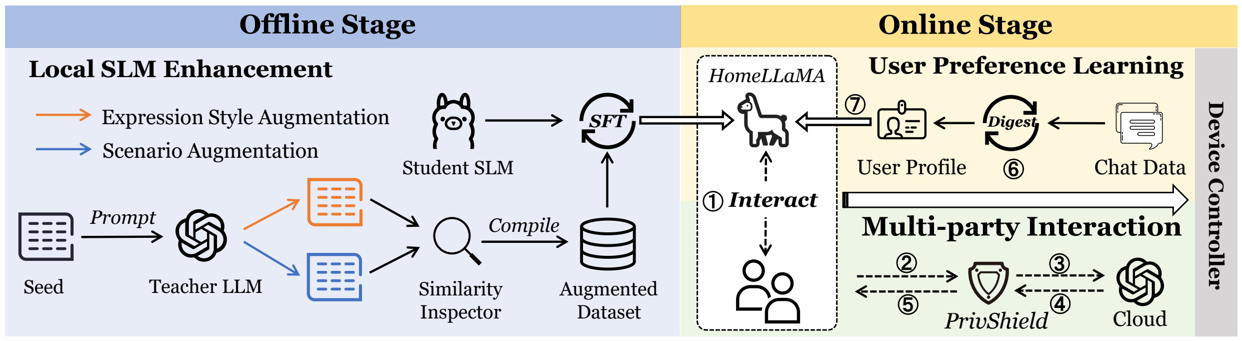


Fig. 3. System overview of *HomeLLaMA*. The system begins with an offline stage to enhance service quality within smart homes. Once deployed, it enters the online stage, where it continuously learns and updates user profiles in real time, with optional cloud assistance upon user request.

LLMs to enhance personalization. However, applying this method to local SLMs faces a unique challenge: local SLMs have a much shorter context length (*e.g.*, only 8K tokens for LLaMA3), making it infeasible to include lengthy user profiles in prompts. While the widely adopted retrieval-augmented generation (RAG) method [23] presents a promising solution for conserving context length by retrieving relevant information from a local database, its long-term use in smart homes may lead to the continuous accumulation of preference-related files. This accumulation can result in increased data redundancy over time, thereby hindering the effective retrieval of highly correlated information. This practical limitation necessitates innovative solutions to optimize the utilization of historical data.

IV. Design of HomeLLaMA

A. System Overview

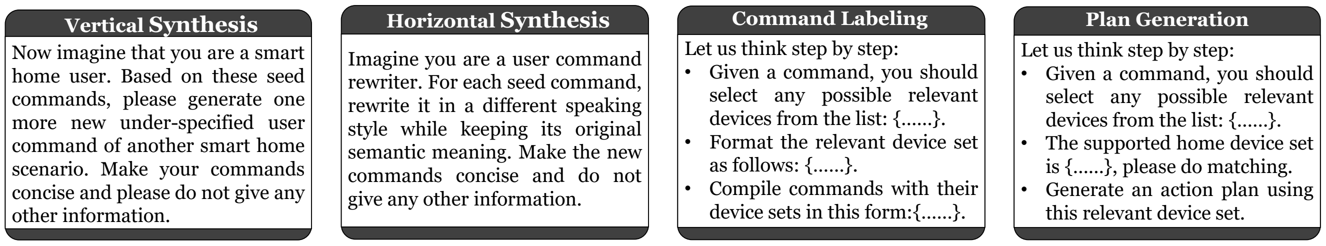
To address the aforementioned challenges, we propose *HomeLLaMA*, with an overview outlined in Fig. 3. Before deployment, it begins with the offline *Local SLM Enhancement* module (§ IV-B) and with the enhanced model, user commands are further processed through the online stage, consisting of the *Multi-party Interaction* module (§ IV-C) and the *User Preference Learning* module (§ IV-D).

- **Local SLM Enhancement** enables the local SLM to generate plans for various user commands. Before deploying the local assistant, it is necessary to enhance the SLM so that it can identify relevant devices based on user commands. We begin by selecting seed commands from an open-source command-action dataset, covering various scenarios such as lighting, environment control, and security [48]. We then propose a tailored data augmentation method by feeding seed commands into a cloud LLM (GPT-4) to generate new commands, incorporating different expression styles (*user diversity*) and scenarios (*application diversity*). This process synthesizes a large set of user commands. Then, the teacher LLM labels the commands with comprehensive relevant devices and compiles them into an augmented dataset, which is further used to fine-tune the local SLM.
- **Multi-party Interaction** further enhances the user experience in the loop of user-assistant-cloud interactions. Users can interact with *HomeLLaMA* by freely expressing their requirements (①). If the response generated by *HomeLLaMA* falls short of expectations, users can give feedback or allow the local assistant to seek advice from cloud LLMs (②). To enhance user privacy, *PrivShield* obfuscates user commands by blending them with adversarial commands generated by the SLM before sending the mixture to a cloud-based LLM for processing (③). The cloud LLM, upon receiving these mixed queries, generates a set of responses and returns them to the local *PrivShield* (④). The real response corresponding to the original user command is then identified and recovered as advice, which the SLM integrates to provide users with a refined action plan (⑤).
- **User Preference Learning** ensures the assistant continuously learns and adapts to user preferences. Specifically, *HomeLLaMA* records each user-assistant interaction, which is then digested into structured user profiles with a predefined format (⑥). And the maintenance of user profiles will dynamically update based on profile similarity. The module allows the assistant to retrieve user profiles to generate personalized plans for similar commands in the future (⑦). Over time, with the accumulated user profiles, *HomeLLaMA* becomes increasingly attuned to user preferences.

B. Local SLM Enhancement

To improve SLMs in smart homes, a viable approach is to apply supervised fine-tuning (SFT) [47] on a tailored dataset mapping ”user command → relevant devices”. Inspired by recent advances in data augmentation methods (*e.g.*, WizardLM [49]), we investigate the potential of leveraging powerful cloud LLMs (*e.g.*, GPT-4) to automatically synthesize a customized dataset with higher quality. This approach effectively transfers the knowledge embedded within the cloud LLM (teacher) to the local SLM (student) through the fine-tuning process.

1) Understanding the dataset: Serving different smart homes with diverse user groups is not a straightforward one-input-to-one-output mapping problem, and two types of diversity need to be considered: **1) Command diversity.** It arises from two main aspects: user diversity and scenario diversity. User diversity refers to the fact that different users may express their requests in various ways, while scenario diversity refers to different types of home scenarios. **2) Device diversity.** It refers to the fact that different smart homes may have varying sets of available devices, leading to multiple possible responses for the same command.



Vertical Synthesis

Now imagine that you are a smart home user. Based on these seed commands, please generate one more new under-specified user command of another smart home scenario. Make your commands concise and please do not give any other information.

Horizontal Synthesis

Imagine you are a user command rewriter. For each seed command, rewrite it in a different speaking style while keeping its original semantic meaning. Make the new commands concise and do not give any other information.

Command Labeling

Let us think step by step:
- Given a command, you should select any possible relevant devices from the list: {......}.
- Format the relevant device set as follows: {......}.
- Compile commands with their device sets in this form:{......}.

Plan Generation

Let us think step by step:
- Given a command, you should select any possible relevant devices from the list: {......}.
- The supported home device set is {......}, please do matching.
- Generate an action plan using this relevant device set.

(a) Vertical synthesis. (b) Horizontal synthesis. (c) Command labeling (cloud). (d) Plan generation (local).

Fig. 4. The prompt template for (a) vertical and (b) horizontal synthesis, (c) command labeling, and (d) plan generation.

2) Command augmentation: Concerning the issue of **command diversity**, it is essential to construct a dataset that includes a wide range of high-quality commands across different user groups and various scenarios. We begin this process by manually selecting a set of under-specified commands as the seed from crowd-sourcing platforms (*e.g.*, IFTTT) [48] based on their popularity, *i.e.*, overall adoption frequency among users. The selected commands encompass several commonly used smart home scenarios, such as climate control and lighting control. Each scenario contains 10 commands, and we obtain a total of 90 seed commands.

Synthesis of new commands. Harnessing the strong generative capabilities of cloud LLMs allows us to expand the dataset without the need for manual data collection. During each iteration of synthesis, we randomly sample five commands from the command pool as a starting point. Motivated by the two aspects of command diversity, we proceed to augment the original commands along the following two directions:

- *Vertical synthesis* generates new commands for different smart home scenarios. With the sampled seed commands, we first instruct GPT-4 to generate a new yet relevant command in a different scenario, using our carefully designed prompts (Fig. 4(a)). With the newly obtained command, we feed it back into GPT-4 to verify whether the command is indeed relevant to the smart home context. If not, we discard the command and proceed to the next iteration.
- *Horizontal synthesis* aims to generate new commands with varied expression styles. Similar to the vertical synthesis process, we instruct GPT -4 (Fig. 4(b)) to modify the expression style of the original command while preserving its original meaning. Once generated, we add the new command to the candidate command pool for further verification.

Similarity inspector. To ensure the quality of augmented commands, it is necessary to remove redundant commands with similar semantic meanings from the candidate pool. Specifically, given any newly generated command s_{new}, let the set of existing commands in the command pool be $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. The ROUGE-L score function, denoted as $R(s, s^{'})$, measures the similarity between commands. Then the retention condition for the new command is

$$\text{Retain } s_{\text{new}} \iff \max_{i\in\{1,2,\ldots,n\}} R(s_{\text{new}}, s_i) < \alpha \tag{1}$$

where α is a predefined threshold that controls the portion of overlap in semantic similarity between the new and existing commands. This means that a new command will be preserved only if the similarity between the new command and any existing command is less than the predefined threshold.

3) Command labeling: Given the augmented command pool, the next critical step is accurately labeling these commands to construct a comprehensive command-device dataset. We leverage the cloud LLM to label the commands with comprehensive device sets, encompassing all potential relevant devices. Specifically, we first simulate a virtual and large-scale smart home deployed with a comprehensive set of COTS devices (39 devices in total) collected from a smart home platform [48]. With the prompt designed in Fig. 4(c), we instruct the cloud LLM to identify a subset of relevant devices from the comprehensive set for each user command in the augmented dataset. The labeling process can be expressed as:

$$\mathcal{D}_a = \{s_i \rightarrow G(s_i, \boldsymbol{D})\},\ \forall s_i \in \mathcal{S}_a \tag{2}$$

where $\mathcal{D}_a$ is the augmented dataset, s_i is a user command from the augmented command pool $\mathcal{S}_a$, $\boldsymbol{D}$ is the comprehensive device set we build, and $G(\cdot)$ represents the black-box LLM.

Remarks. For the uncommon situation where a smart home contains a device not included in the comprehensive set, the user can explicitly suggest the missing device and specify their preference on how to adjust it. Such explicit user feedback will be recorded and retrieved for future reference § IV-C. Note that the labeling process is agnostic to distinct device configurations and does not require transmitting specific user data to the cloud LLM.

4) Training the adapter as the device identifier: After obtaining the tailored command-device dataset, we proceed to fine-tune the local SLM to enhance its capabilities. Specifically, we utilize the QLoRA technique [50], a widely adopted parameter-efficient fine-tuning (PEFT) [51] method. Instead of fine-tuning all model parameters, which is both resource-intensive and time-consuming, QLoRA trains a lightweight adapter integrated into the target model. In the smart home context, this process involves training a LoRA adapter to act as a device identifier for the local SLM. By combining this adapter with the original SLM, which retains extensive world knowledge, *HomeLLaMA* becomes more adept at accurately identifying and interacting with various smart devices.

5) Inference paradigm: With the enhanced SLM, we then propose a tailored inference paradigm for serving each individual home concerning the **device diversity** based on Chain-of-Thoughts (CoTs) [52]. Fig. 4(d) illustrates our designed prompt that instructs the SLM to generate the corresponding plans in

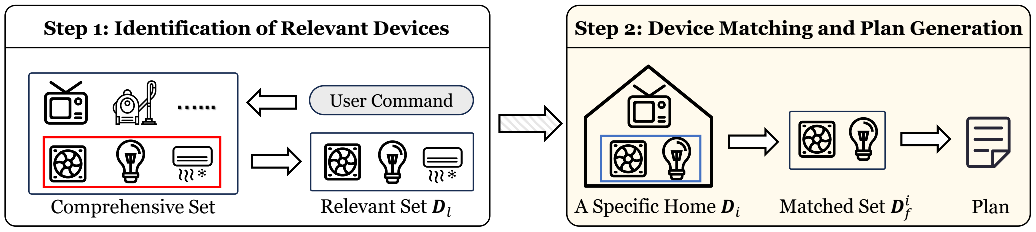


Fig. 5. The designed inference paradigm of the local SLM.

a step-by-step manner. The inference paradigm can be divided into two steps, outlined in Fig. 5:

- Initially, we consider a large home equipped with almost all COTS devices, as mentioned before. Then, we prompt SLM to generate a comprehensive list of relevant devices given a command. We denote the comprehensive relevant device set as $\boldsymbol{D}_l$, as shown in the red box of Fig. 5.
- Then, the generated results are adapted to a specific home by performing a matching process. Specifically, with the available device set in home i denoted as $\boldsymbol{D}_i$, we prompt the SLM with the instructions in Fig. 4(d) to execute the task, matching the common devices of $\boldsymbol{D}_l$ with $\boldsymbol{D}_i$ to obtain the matched set $\boldsymbol{D}_f^i$ for home i (as shown in the blue box of Fig. 5). The matching operation via the SLM is:

$$\boldsymbol{D}_f^i = \boldsymbol{D}_l \cap \boldsymbol{D}_i. \tag{3}$$

Remarks: In the initial step, while it is feasible to directly input the device set of a target home to generate the action plan, this approach may result in performance degradation. The main reason is that the local SLM is fine-tuned on our tailored dataset with a predefined input format. Therefore, the enhanced ability in relevant device identification may only be activated when the prompt aligns with the expected format.

C. Multi-party Interaction

As mentioned in § III-B, the enhanced SLM may still fail to consistently offer high-quality services in practice. To further enhance the user experience, we propose a *multi-party interaction* module that facilitates user feedback and consultation with cloud LLMs when necessary. This module supports two types of interactions: ❶ the user-assistant interaction, which allows users to explicitly specify their requests and preferences; and ❷ the user-driven local-cloud collaboration, where the local SLM is triggered by users to consult cloud LLMs for improved services with enhanced privacy protection.

1) User-assistant interaction: As the core component of *HomeLLaMA*, the user-assistant interaction acts as an interface for users to express their intents. In the flowchart illustrated in Fig. 6, upon receiving a user command, the assistant generates action plans and responds to the user for confirmation. For every generated response to users, they may accept, reject, or follow up with a piece of advice.

- **Accept:** If the user is satisfied with the generated action plan, the action plan will be translated into the command to smart devices to control relevant devices.
- **Advice:** The user can provide natural language feedback to further refine the user intent. For example, suppose the user inputs a command like ”Brighten the bedroom,” and the assistant responds with ”Turn on all the lights in the bedroom.” If the user only wants to turn on the bedside lamp, she can follow up with a detailed instruction (*e.g.*, ”Bedside lamp only, please.”). The assistant will then regenerate the action plan by incorporating the user’s advice.
- **Reject:** If the user is not satisfied with the local response, she may reject the action plan. For instance, if the user wants to hold a home party and inputs ”Let the party begin,” but the assistant responds with a simple action like ”Turn on the lights and adjust the room temperature,” the user might reject the response. In that case, the assistant leverages the cloud LLM to generate an improved action plan with the user-driven local–cloud collaboration module (§ IV-C2).

Remarks. Note that after each generated response, including revised plans resulting from ”Advice” or ”Reject,” the user can further interact with the assistant. Only when the user explicitly ”Accepts” a proposed action plan will the plan be translated to control smart devices accordingly (Fig. 6). Subsequently, the command and the final approved plan will be saved as an interaction record, which will be further utilized by the preference learning module (§ IV-D).

2) Local-cloud collaboration: When the user rejects a plan, *HomeLLaMA* will ask the user for permission to consult a cloud model (*e.g.*, GPT-4). If approved, the assistant will proceed with generating an improved response.

Potential privacy risks. However, directly querying the cloud LLM via registered API calling with the raw user command and home details may raise privacy concerns since user activities and personal information may be inferred and monitored indirectly (*e.g.*, through differential attacks [53]) by the curious cloud servers. For example, suppose a user first requests, ”At 9 pm, make my living room chilly and turn on the TV,” followed by, ”At 10 pm, check if the doors and windows are locked and make my bedroom comfortable.” From these commands, it can be easily inferred that the user might be watching TV from 9 pm to 10 pm and then go to bed.

Role of *HomeLLaMA* during collaboration. While the goal of local–cloud collaboration is to enhance user experience, it must not come at the cost of unacceptable privacy compromises. To this end, *HomeLLaMA* functions as both a processing center and a privacy guardian: it retains all smart home configurations and user profiles locally, transmitting only the current user command to the cloud for assistance. To further mitigate potential privacy risks inherent in raw commands, we incorporate *PrivShield*, a lightweight obfuscation module designed to obscure user queries, as shown in Fig. 7.

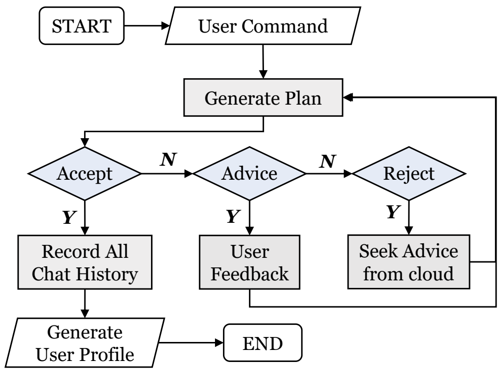


Fig. 6. The user-assistant interaction flow.

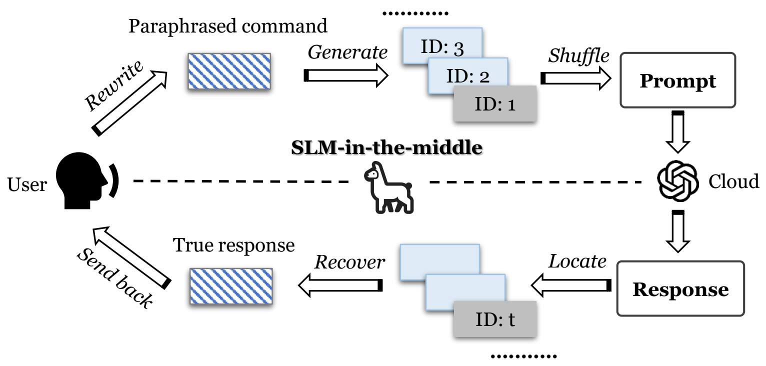


Fig. 7. Workflow of the *PrivShield*.

PrivShield. Essentially, the *PrivShield* operates within a **SLM-in-the-middle** framework. In practice, *PrivShield* safeguards user privacy through procedures including user command rewriting, adversarial command generation, and plan recovery.

- *User command rewriting.* An original user command may contain personal information (*e.g.*, names, locations) and many colloquial expressions (*e.g.*, modal particles). These components not only introduce information redundancy but also provide opportunities for third parties to infer the user's actual command through continuous pattern recognition in subsequent processes. To address this, we direct the local SLM to first filter sensitive personal information [54], and then paraphrase the original user command using the customized prompt illustrated in Fig. 8(a).
- *Adversarial command generation.* Given a paraphrased command, the *PrivShield* prompts the local SLM with the designed instructions in Fig. 8(b) to generate other N adversarial commands across various unrelated scenarios to obscure the original command. Each of the commands is assigned a unique command ID and shuffled, with only the original command's ID t being locally recorded. These commands are subsequently combined into a single query along with their respective command IDs, as shown in Fig. 8(c). The combined query will be transmitted to a cloud LLM to generate action plans for all the commands.
- *Action plan recovery.* Upon receiving the response from the cloud LLM, the *PrivShield* extracts the comprehensive action plan associated with the right order. This extracted action plan is then fed into the local assistant as advice for generating a tailored plan for the user. The tailored plan is subsequently delivered to the user as the updated plan.

Remarks. *PrivShield* enables users to access cloud services with privacy protection in an easily understandable manner. However, the assistant primarily operates locally whenever possible. The reasons are twofold: 1) User profiles and home configurations are stored locally and will not be transmitted to the cloud for processing due to privacy concerns. 2) The cost of constantly querying the cloud may be prohibitive. Before deployment, users are allowed to customize the number of adversarial commands, *i.e.*, N, to achieve user-oriented privacy-cost balance as discussed in § VIII.

D. User Preference Learning

Due to the restricted context length and information redundancy, local SLMs cannot simply store all the chat history for generating personalized responses. To address this challenge, we develop a lightweight user profiling method, enabling the assistant to efficiently retrieve a dynamically updating user profile database for reference. In practice, the user preference learning module operates in three key stages: user profile generation, profile updating, and personalized plan generation.

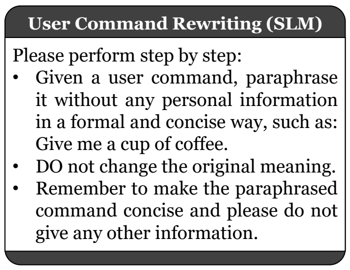
1) User Profile Generation: The interaction records between the user and the assistant are locally recorded. A structured prompt, as illustrated in Fig. 9(a), guides the local SLM to digest and generate a well-organized user profile for each conversation. These profiles include details on ① **topics** (*i.e.*, the keywords of conversations summarized by the SLM), ② **preferences**, ③ **commands**, and ④ **final action plans** in a concise way. These profiles are then transformed into vector representations and stored in a text embedding database $\mathcal{E}$.

2) Profile Updating: The user profile database follows a carefully designed updating mechanism to maintain its effectiveness over time. When a new user profile is generated, it is compared with all existing profiles via cosine similarity. If the similarities between the new profile and all the existing profiles are below a pre-defined threshold, the new profile will be saved as a distinct entry in the database. Otherwise, it is constructively merged with the most similar existing profile. Specifically, given the embedding of a newly generated user profile denoted as p_n, the condition for inserting this profile into the embedding database is determined by:

$$\text{Insert } p_n \text{ into } \mathcal{E} \iff \max_{\forall p_i \in \mathcal{E}} C(p_\text{n}, p_i) < \beta \tag{4}$$

where $C(\cdot)$ is the cosine similarity function and β is a pre-defined similarity threshold. If the maximum cosine similarity between p_n and any existing profile p_i in the database is less than β, the new profile will be inserted as a distinct entry. Otherwise, the two similar profiles are merged into a new, consolidated profile, which replaces the original profile by prompting the SLM with the prompts in Fig. 9(b).

3) Personalized Plan Generation: During the inference stage, given a new query q_i, the assistant retrieves the top-3 user profiles in the form of text embedding (denoted as p_m, p_n, and p_p) that have the highest cosine similarity to the query.

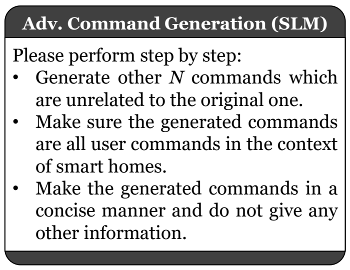


User Command Rewriting (SLM)

Please perform step by step:

- Given a user command, paraphrase it without any personal information in a formal and concise way, such as: Give me a cup of coffee.
- DO not change the original meaning.
- Remember to make the paraphrased command concise and please do not give any other information.

(a) User command rewriting (SLM).

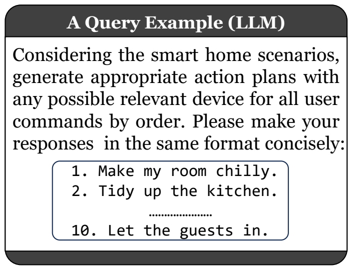


Adv. Command Generation (SLM)

Please perform step by step:

- Generate other N commands which are unrelated to the original one.
- Make sure the generated commands are all user commands in the context of smart homes.
- Make the generated commands in a concise manner and do not give any other information.

(b) Adv. command generation (SLM).



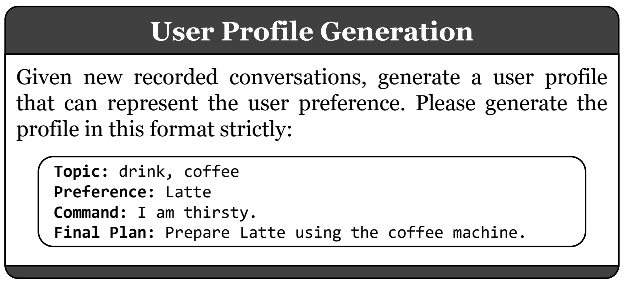
A Query Example (LLM)

Considering the smart home scenarios, generate appropriate action plans with any possible relevant device for all user commands by order. Please make your responses in the same format concisely:

```
1. Make my room chilly.
2. Tidy up the kitchen.
......................
10. Let the guests in.
```

(c) A query example (LLM).

Fig. 8. The prompt templates of the designed *PrivShield*.

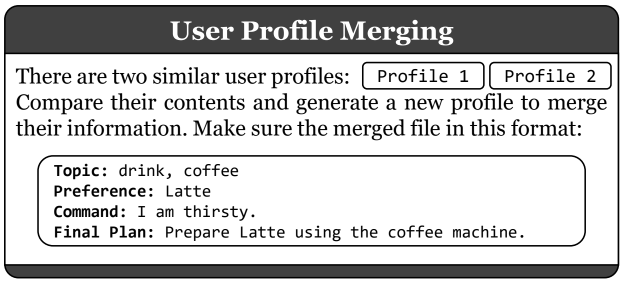


User Profile Generation

Given new recorded conversations, generate a user profile that can represent the user preference. Please generate the profile in this format strictly:

```
Topic: drink, coffee
Preference: Latte
Command: I am thirsty.
Final Plan: Prepare Latte using the coffee machine.
```

(a) User profile generation.



User Profile Merging

There are two similar user profiles: `Profile 1` `Profile 2` Compare their contents and generate a new profile to merge their information. Make sure the merged file in this format:

```
Topic: drink, coffee
Preference: Latte
Command: I am thirsty.
Final Plan: Prepare Latte using the coffee machine.
```

(b) User profile merging.

Fig. 9. The prompt templates for user profile generation and merging.

We then convert the selected embedding into text (denoted as u_m, u_n, and u_p) through decoding:

$$(p_m, p_n, p_p) \xrightarrow{\text{Decode}} (u_m, u_n, u_p) \tag{5}$$

By concatenating the decoding result with the user query q_i, the personalized output plan is generated based on these profiles and the current home configuration H_i:

$$\text{Plan} \leftarrow \mathcal{L}(u_m, u_n, u_p, H_i, q_i) \tag{6}$$

where $\mathcal{L}$ represents the action plan generation function of the local SLM. The generated action plan is subsequently used to control the corresponding smart devices.

Remarks: The user preference learning module can be extended to multi-user scenarios by constructing separate databases. During service, the assistant will first perform voice recognition to determine the user's identity before processing.

V. Implementation

We implement *HomeLLaMA* on a local server. The implementation details of each key component are as follows:

Data Augmentation: We prepare seed commands from IFTTT and access OpenAI's services via the OpenAI API. During the augmentation process, we select GPT-4-Turbo as the default LLM and appropriately prompt it to generate the required data. By default, we set α in Equation 1 to 0.7. The augmented dataset contains 14K action-command pairs in total, covering 9 common smart home scenarios (*e.g.*, atmosphere adjustment, power management, *etc*).

Fine-Tuning: We choose Meta-LLaMA3-8B [55] as our base model downloaded from Hugging Face [4]. To fine-tune the base model with our augmented dataset, we utilize the QLoRA [50] technique with 8-bit quantization. The rank r is set to 64 and $lora\_alpha$ to 128. The learning rate is initialized at 3×10^{-5}, and a dropout rate of 0.1 is applied to alleviate the over-fitting problem. The number of fine-tuning epochs is 3, with an 80-20 train-test split ratio. The model is fine-tuned on a server running Ubuntu 22.04 LTS with a single NVIDIA RTX 4090 GPU, taking approximately 8 hours.

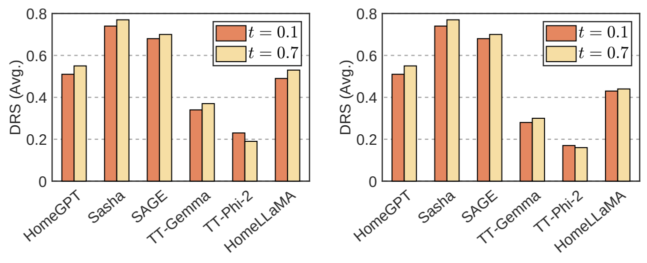
User Profile Database: We deploy and maintain the user profile database using FAISS [56], a library for efficient similarity search and clustering of dense vectors. The conversation histories collected from the interactions are saved in the text format and summarized into user profiles. Those well-summarized user profiles are stored in the text embedding database and are ready to be retrieved for the generation of personalized plans during the inference stage. By default, the β in Equation 4 is set to 0.6.

VI. Performance Evaluation

In this section, we conduct comprehensive, quantified experiments to evaluate the effectiveness of *HomeLLaMA* in addressing the performance-privacy dilemma. Specifically, we aim to answer the following questions:

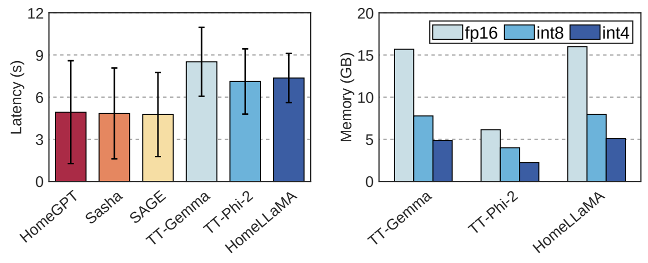
- **Q1** - *Performance*: Can *HomeLLaMA* provide high-quality services locally?
- **Q2** - *Privacy*: Does *HomeLLaMA* quantitatively enhance user privacy?
- **Q3** - *System Overhead*: Is *HomeLLaMA* affordable to be deployed locally?
- **Q4** - *Sensitivity*: How do system configurations (*i.e.*, base models) impact performance?

[4]https://huggingface.co



(a) FP16 precision of SLMs. (b) INT8 precision of SLMs.

Fig. 10. Avg. *DRS* after setting (a) FP16 and (b) INT8 precision.



(a) Response latency. (b) GPU memory usage.

Fig. 11. Results for (a) latency and (b) memory usage.

A. *Model Capacity (Q1)*

1) DevFinder Benchmark: Considering the comprehensive home setup described in § IV-B, which is equipped with commonly used smart devices, we select 100 test commands with human-annotated device labels from the IFTTT dataset [48] with more details outlined in the open-source dataset. These commands with labels encompass a wide range of scenarios, including environmental control, atmosphere adjustment, power management, *etc*. The commands are input into the smart home assistants to generate responses, which are then compared with the annotated labels to quantify the quality of the generated action plans.

To quantify *HomeLLaMA*'s capability in identifying relevant devices, we adopt the Device Relevance Score (*DRS*) as the evaluation metric [7]. Suppose the ground truth device set is G_l, and the device set generated by the local SLM is G_r, we compute the relevance score as:

$$DRS = \frac{|G_l \cap G_r| - |G_r - G_l|}{|G_r|} \tag{7}$$

where $|G_l \cap G_r|$ represents the number of overlapping devices, and $|G_r - G_l|$ refers to the number of devices in the response that are not included in the ground truth. Then, the relevance score is normalized to $[-1, 1]$. The higher the relevance score, the better the performance in identifying relevant devices.

2) Baselines: To contextualize the model performance of *HomeLLaMA*, we compare our system with several other LLM-powered smart home baselines:

- **HomeGPT** [24] directly prompts the LLM to generate smart home plans in response to user commands.
- **Sasha** [7] modifies HomeGPT by introducing a revised pipeline consisting of five procedures, including filtering, planning, *etc*, to enhance the quality of plans.
- **SAGE** [22] generates personalized plans by inserting all conversation history into prompts.
- **Thoughtful Things** (**TT**) [25] utilizes the on-device SLMs (Google Gemma-7B [57] and Microsoft Phi-2-3B [58]) to generate routines and control smart devices.

Note that for a fair comparison, the cloud-assisted module is disabled during the evaluation of device relevance, and all results are computed purely based on local operations.

3) Results: We test *HomeLLaMA* and baselines on the proposed *DevFinder* and report the average score of each system. Additionally, we set the temperature t of these models to 0.1 and 0.7 for evaluation, respectively. As illustrated in Fig. 10, the designed *HomeLLaMA* significantly outperforms the other two on-device assistants but still lags behind cloud-based LLM assistants. To investigate the reasons behind this, we examine and analyze the generated action plans and uncover the following insights:

- Despite the superior performance of cloud-based assistants enabled by larger models [21], *HomeLLaMA* achieves comparable *DRS* to GPT-3.5 while ensuring user privacy through *PrivShield* and local processing, demonstrating high performance without compromising user privacy.
- Among on-device assistants, *HomeLLaMA* delivers the best local service, outperforming TT-Gemma and TT-Phi-2 in device relevance. This is due to its fine-tuning on smart home–specific data and its hybrid design, which enables selective cloud assistance to further boost *DRS* when needed.

Prompt template for Launching Activity Monitoring Attacks

You are now playing the role of an attacker attempting to uncover users' in-home activities. You will be presented with a set of commands, among which only one is the true user command. Note that human-generated commands may differ significantly from machine-generated ones. Specifically, your task is to:
- Identify user commands with the highest likelihood of being correct.
- As you gather more prompts across multiple rounds, try to recognize the patterns in user commands, and utilize pattern matching to improve your accuracy in the subsequent identifications.

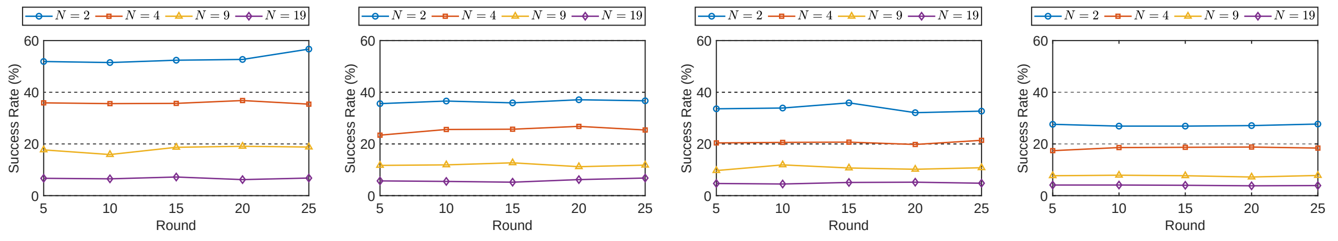
Fig. 12. The prompt template for launching activity monitoring attacks.

B. *Privacy Protection (Q2)*

1) Qualitative risk analysis: Before quantifying *HomeLLaMA*'s privacy protection, it is important to first examine potential privacy risks qualitatively. In typical cloud-based smart home systems, users must register for API keys and transmit commands and device states to remote servers, exposing them to risks during data storage, network transmission, and inference, as detailed in Table II. These include unauthorized access, interception, and inference attacks such as PII extraction, attribute inference, and activity monitoring. Unlike such systems, *HomeLLaMA* operates locally without sharing user profiles or home configurations, thereby mitigating storage and transmission risks. Additionally, its *PrivShield* module filters sensitive content before sending prompts to the cloud, preventing PII exposure. However, obfuscated prompts remain susceptible to activity monitoring via API traceability. The next section presents quantitative experiments to evaluate resilience against this residual threat.

TABLE II
QUALITATIVE RISK ANALYSIS OF LLM-BASED ASSISTANTS INCLUDING CLOUD-BASED SYSTEMS, LOCAL TT, AND *HOMELLAMA*. HERE, "●" INDICATES COMPROMISING PRIVACY REGARDING THIS THREAT, WHILE "◐" SIGNIFIES IT REQUIRES QUANTITATIVE EVALUATION (§ VI-B2).

Threat Type		Cloud-Based Systems	TT (Local)	HomeLLaMA (Hybrid)
Data Storage		●	○	○
Network Transmission		●	○	○
Inference [40]	PII Extraction [59]	●	○	○
	Attribute Inference [60]	●	○	○
	Activity Monitoring [46]	●	○	◐



(a) Phi3-3B & GPT4-attacker. (b) Llama3-8B & GPT4-attacker. (c) Llama3-13B & GPT4-attacker. (d) Llama3-13B & classifier-attacker.

Fig. 13. privacy risks across query rounds and local SLMs for different N, with a GPT-based attacker from (a) to (c), and (d) a classifier-based attacker.

2) Quantitiave Analysis: We focus on examining the in-home activity monitoring threat in *HomeLLaMA*. During the use of *PrivShield*, the real commands are obfuscated with N other SLM-generated adversarial commands before being sent to the cloud LLM for processing.

Threat Model. We assume the cloud LLM operates on the *honest-but-curious* remote server [31], where adversaries deliver correct inference results but investigate all transmitted user queries. Under our framework, the adversary's goal is to identify the real query from the mixture, thereby enabling real-time monitoring of users' in-home activities. Specifically, following procedures in [61], we launch this *activity monitoring attack* by using ❶ a cloud GPT-4 with prompts shown in Fig. 12, and ❷ a well-trained text classifier [62] fine-tuned on *DevFinder*, to infer the user's true command. We then use *attack success rate* [46] to assess the system's privacy level, where a higher rate indicates a greater threat to user privacy and reduced system protection.

Results. We use the constructed *DevFinder* as test user queries, setting the number of adversarial commands N to 2, 4, 9, and 19. We also vary the base models (Phi3, LLaMA3-8B & 13B) to examine their impacts on the quality of adversarial command generation. The attack success rates across distinct conditions obtained through extensive experiments are presented in Fig. 13, and we report the following findings:

- *PrivShield* effectively safeguards user privacy by maintaining significantly lower attack success rates compared to direct queries without its protection. As shown in Fig. 13, with different attacker models (GPT-4 and a well-trained classifier), leveraging *PrivShield* with different N values and various base models results in a substantial reduction in attack accuracy, far below the 100% success rate of direct queries. Furthermore, while an increase in query rounds allows adversaries to accumulate more information, as illustrated in all sub-figures, this does not enhance their ability to accurately infer user prompts. In fact, denote the average attack success rate of the *PrivShield* as SR_p, and the overall attack success rate SR_h should be

$$\mathrm{SR}_h = \epsilon \cdot \mathrm{SR}_p \tag{8}$$

where ϵ represents *PrivShield*'s frequency of use. During practical daily usage, the frequency ϵ will gradually become lower with user profiles being progressively constructed, as discussed in § VII-B. Consequently, the overall privacy protection will strengthen over time, as users will increasingly rely on local operations without cloud assistance.
- Privacy protection strengthens as the number of adversarial commands increases. These adversarial commands can **introduce noises in the text space**, making it harder for malicious attackers to identify the real queries. Moreover, generating more adversarial commands incurs higher latency and cost, users should be allowed to decide their preferred trade-off between privacy and performance.
- Privacy protection also benefits from the use of stronger SLMs. As demonstrated in figures, replacing base SLMs with stronger models significantly reduces attack accuracy. This is because **identifying real user queries becomes equivalent to distinguishing AI-generated text from the mixture**. Stronger SLMs are more adept at generating high-quality adversarial commands, further obscuring the real query from identification. However, deploying larger models may be impractical due to resource constraints, which will be discussed further in § VI-D.

Remarks. Although adversaries may deploy advanced pre-trained classifiers to distinguish user commands from obfuscated mixtures. In such cases, *PrivShield* could be enhanced by strengthening query obfuscation (*e.g.*, selecting a larger N) for stronger privacy protection according to users' requirements.

C. System Cost (Q3)

1) Metrics: We evaluate the system cost in terms of the following aspects: **1) Response latency:** We measure the time

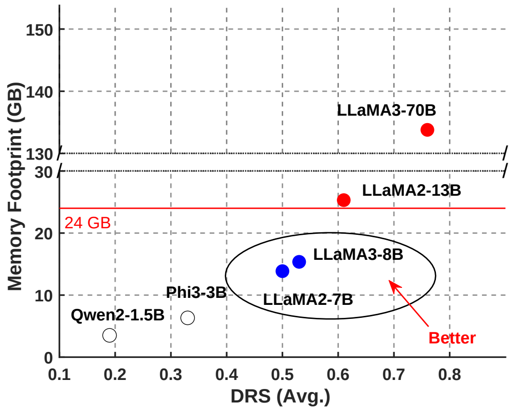


Fig. 14. Impacts of different base models.

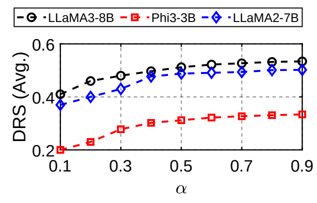
cost from the moment the user inputs a command to the generation of the final action plan as the response latency of each system. **2) Memory usage:** We measure the system overhead by tracking the GPU memory usage (in GB) via a Python package named memory-profiler [63] during usage.

2) Baselines: We keep the same baselines as selected in § VI-A2. Note that the first three systems (*i.e.*, HomeGPT, Sasha, and SAGE) are based on cloud LLMs, which cannot be directly accessed, while only TT explores the integration of SLMs into smart home assistants. Consequently, we only measure the memory usage of TT-Gemma, TT-Phi-2, and our proposed *HomeLLaMA*.

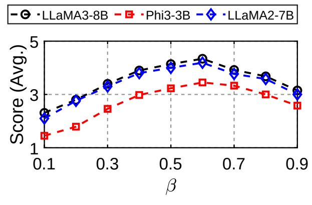
3) Results: To measure the system overhead of *HomeLLaMA* and the baselines, we input each test command in the *DevFinder* benchmark into them and report the average response latency with its variance. As shown in Fig. 11(a), the cloud-based assistants exhibit relatively faster average response time (around 4.97 seconds) compared to the local-based assistants, primarily due to the performance optimizations of OpenAI services [13]. However, cloud-LLM-based systems are highly susceptible to network conditions and server stability, leading to significant variance in response latencies, which can negatively impact user experience. In contrast, *HomeLLaMA* exhibits less variance in response latency, albeit with a slightly longer response time.

We further track the maximum memory usage of the two local SLM-based systems (*i.e.*, TT and *HomeLLaMA*) when adopting different quantization precisions (*i.e.*, fp16, int8, and int4) during inference. As shown in Fig. 11(b), the maximum GPU memory requirement for these systems remains under 16 GB, which is affordable and manageable for a typical household[5]. Notably, *HomeLLaMA* is not restricted to GPU-only deployment: according to the literature [64], with INT8 quantization, the backbone model (*e.g.*, LLaMA3-8B) can be executed entirely on CPU-only platforms, with a practical memory consumption of approximately 12–16 GB of system RAM under a 4K-token context length, which is well within the capability of many modern household desktops, mini-PCs, and other devices, enabling deployment in GPU-free environments while still maintaining stable inference performance.

[5]An NVIDIA RTX 4070 Ti SUPER GPU (16 GB) costs around $840.



(a) Distinct α for the inspector.



(b) Distinct β for merging profiles.

Fig. 15. Impacts of selecting different thresholds α and β.

D. Sensitivity Analysis (Q4)

1) Different base models: We evaluate the impact of base models by deploying Qwen2-1.5B [65], Phi3-3B [66], LLaMA2-7B/13B [67], LLaMA3-8B [55], and LLaMA3-70B [67] in *HomeLLaMA*. Given the test inputs from *DevFinder*, we record their average *DRS* and GPU memory usage. As shown in Fig. 14, models fall into three groups: (1) lightweight models (white) such as Qwen2-1.5B and Phi3-3B offer minimal memory usage ($\sim$8GB) but poor performance; (2) high-end models (red) like LLaMA2-13B, LLaMA3-70B yield the best *DRS* but require $\geq$24GB GPU memory; (3) mid-range models (blue) such as LLaMA2-7B and LLaMA3-8B balance performance and cost. Thus, we choose LLaMA3-8B as the default base model.

2) Different α during augmentation: To study the effect of the augmentation threshold α (Eq. 1), we vary it from 0.1 to 0.9 and fine-tune LLaMA3-8B, Phi3-3B, and LLaMA2-7B on corresponding augmented datasets. As shown in Fig. 15(a), increasing α improves average *DRS* by promoting data diversity. However, the gain saturates at higher values, while training cost rises due to dataset expansion. We therefore set $\alpha = 0.7$ by default.

3) Different β during profile updating: The profile update threshold β (Eq. 4) determines when to save a new user profile. Using 10 participants, we evaluate personalization ratings (1–5) under varying β from 0.1 to 0.9. As shown in Fig. 15(b), higher β initially improves satisfaction by preventing premature profile merging, but overly high values lead to redundant entries, impairing retrieval. Satisfaction peaks around $\beta = 0.6$, which we adopt as the default.

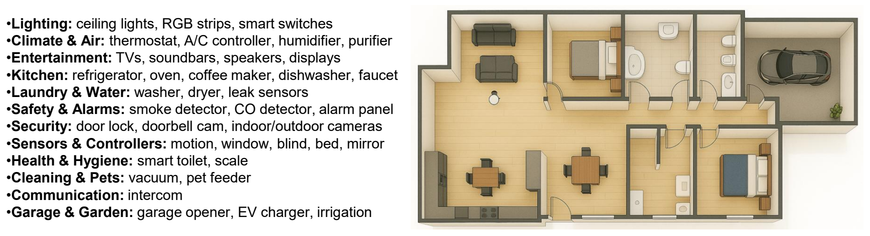
VII. User Study

We conduct user studies to gather feedback from users. The study can be divided into two parts: an *online survey* and an *onsite interview*, aiming to answer those questions:

- **Q5** - *User Privacy Confidence:* Does *HomeLLaMA* lift user-perceived privacy confidence?
- **Q6** - *User Satisfaction:* How do users feel about the overall service quality of *HomeLLaMA*?
- **Q7** - *Long-term Personalization:* Is *HomeLLaMA* capable of adapting to users' preferences continuously?

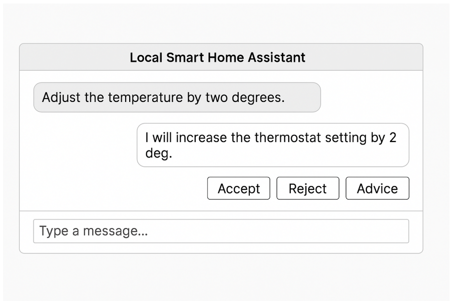
A. Online Survey: Cold-start Evaluation (Q5 & Q6)

1) Preparation works: We select two representative baselines: a cloud-based assistant SAGE and a local-based assistant TT-Gemma. For each scenario, we select a test command



- **Lighting:** ceiling lights, RGB strips, smart switches
- **Climate & Air:** thermostat, A/C controller, humidifier, purifier
- **Entertainment:** TVs, soundbars, speakers, displays
- **Kitchen:** refrigerator, oven, coffee maker, dishwasher, faucet
- **Laundry & Water:** washer, dryer, leak sensors
- **Safety & Alarms:** smoke detector, CO detector, alarm panel
- **Security:** door lock, doorbell cam, indoor/outdoor cameras
- **Sensors & Controllers:** motion, window, blind, bed, mirror
- **Health & Hygiene:** smart toilet, scale
- **Cleaning & Pets:** vacuum, pet feeder
- **Communication:** intercom
- **Garage & Garden:** garage opener, EV charger, irrigation

Fig. 16. The experimental smart home layout with 39 listed devices.



Local Smart Home Assistant

Adjust the temperature by two degrees.

I will increase the thermostat setting by 2 deg.

Fig. 17. GUI interface for participants.

from *DevFinder* and input it into systems to generate initial action plans. Subsequently, we manually select the ”Advice” option (§ IV-C1) and provide the feedback to all systems. These systems then perform preference learning, if applicable, and regenerate action plans that are recorded for further analysis. To ensure fairness, the order of system presentation is randomized before being rated by participants.

2) Participants: We created an online survey using Microsoft Forms and distributed it via email and social media to recruit participants. Before participating, all respondents were presented with an informed consent form outlining the purpose of the study, data handling procedures, and their rights as participants. The study protocol was reviewed and approved by our institution’s ethics review board (IRB). Participants were informed that their responses would be anonymized and used solely for academic research purposes. Participation was entirely voluntary, and no monetary compensation was provided. After 12 days, we received **100 responses** from volunteers in total, and the data of participants shows a relatively balanced distribution in terms of gender, age, educational background, English proficiency, and familiarity with smart assistants.

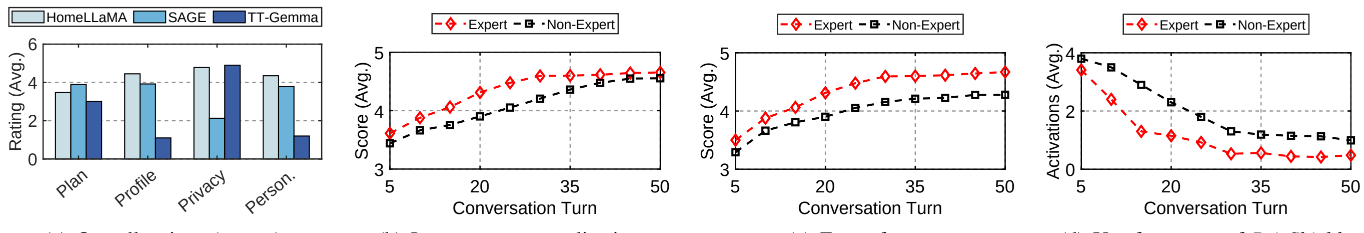
3) Survey design: The survey evaluated smart home systems from the following four metrics: ① **General Plan Satisfaction**, assessing satisfaction with the quality of initial action plans irrespective of personalization; ② **User Profile Correctness**, measuring how accurately the generated user profile reflects personal preferences and behaviors during the current interaction; ③ **Personalization Fit Score**, evaluating how well the responses and recommendations align with historical feedback within the interaction round; and ④ **Privacy Assurance Score**, gauging participants’ confidence in the system’s ability to safeguard personal data and maintain privacy. Participants were asked to rate these metrics on a Likert scale [68] from 1 (*e.g.*, not at all) to 5 (*e.g.*, completely).

4) Overall results: Fig. 18(a) visualizes the overall rating results, and key observations of each aspect include: 1) Participants expressed high satisfaction (close to cloud-based assistant SAGE) with the quality of services provided by *HomeLLaMA*, reflecting its ability to deliver effective and reliable action plans tailored to user needs. This may be attributed to the tailored enhancement method of SLMs proposed in this paper. 2) *HomeLLaMA* excelled in delivering personalized responses and generating precise user profiles compared with other baselines, achieving the highest average score of around 4.35 points. This may be attributed to the *User Preference Learning* module, which accurately characterizes well-structured user profiles to adapt plans effectively. 3) *HomeLLaMA* received high ratings for its privacy-preserving features, surpassing cloud-based solutions and approaching the level of the fully local system, TT-Gemma. This demonstrates its ability to enhance user-perceived privacy by operating locally and minimizing data transmission to the cloud. Additionally, the local-cloud collaboration paradigm, supported by the designed *PrivShield*, boosts users’ confidence in privacy, thereby improving the overall usability of the system.

B. Onsite Interview: Long-term Evaluation (Q7)

To evaluate the long-term performance of *HomeLLaMA*, we deployed the system locally to conduct an on-site interview, following the implementation details outlined in § V. Specifically, we implemented the local smart home assistant on the experimental PC, along with a custom-designed graphical user interface (GUI) (as shown in Fig. 17) that provides users with three options: accept, reject, or request advice as introduced in § IV-C. The smart home layout used in this on-site study is depicted in Fig. 16, comprising a total of 39 commonly used smart devices. The interview spanned 25 days and included 50 conversation turns, with volunteers using the system for an average of approximately 45.6 minutes. Each turn represents a complete user–assistant–cloud interaction (Fig. 6), beginning with a user command and concluding with the execution of the final action plan.

1) Procedures: We deployed *HomeLLaMA* on a laboratory PC using the configuration in § V. Of the 100 survey respondents described in § VII-A1, 10 participants were invited and divided into two groups: 5 experts and 5 non-experts, based on their familiarity with smart homes. Prior to interviews, all participants provided informed consent and received compensation in the form of supermarket coupons valued at approximately \$10. Participants were first introduced to the interaction workflow, including the accept, advise, and reject options, to ensure familiarity. Additionally, two evaluation metrics were explained to them before beginning the interaction: ① **Long-term Personalization** assesses how effectively the system infers and retains user preferences over time. ② **Ease of Use** evaluates how efficiently the system minimizes user efforts for delivering satisfactory responses as usage continues. Participants were then invited to freely interact with *HomeLLaMA*, issuing smart home commands in natural language through UI without constraints. After every



(a) Overall ratings (survey). (b) Long-term personalization. (c) Ease of use. (d) Use frequency of *PrivShield*.

Fig. 18. User study results, including (a) online survey ratings, and (b)–(d) show average interview results.

five conversation turns, they rated the system using the two predefined metrics on a 5-point scale. Throughout the session, we recorded all evaluation scores and, every five turns, also tracked the number of times *PrivShield* was activated for cloud assistance. Finally, the average results for each group were computed and analyzed separately.

2) Results: As illustrated in Fig. 18, both expert and non-expert participants show a steady increase in ratings for *HomeLLaMA* across personalization and ease of use as the number of conversation turns grows, reflecting the system's ability to adapt through dynamically maintained user profiles. Experts tend to assign higher scores earlier, likely due to their clearer articulation of preferences, which accelerates the quality of profiles and system adaptation. A consistent gap remains in ease-of-use ratings, with experts maintaining an advantage of about 0.35 points by the 50th turn, though both groups converge above 4.2, indicating strong usability. Additionally, the activation frequency of *PrivShield* declines for both groups, approaching zero by the 50th turn, highlighting *HomeLLaMA*'s reduced reliance on cloud-based support as it better internalizes user preferences, thereby enhancing privacy protection.

VIII. Discussion

***PrivShield* customization.** As described in the design of *PrivShield* (§ IV-C2), users can customize the number of generated adversarial commands, denoted as N, prior to deployment. A larger value of N offers stronger privacy protection but also increases system overhead, including processing latency and command token usage. To assist users in selecting an appropriate setting, we provide three representative configurations of N, *i.e.*, $N = 2, 4, 9$, corresponding to low, medium, and high levels of privacy protection. The associated privacy risks and system costs of *PrivShield* for each setting are summarized in Table III, based on average results over 100 randomly selected user commands. As shown in Table III, while higher values of N provide stronger privacy guarantees, they also introduce additional system overhead, with token usage increasing from 33.2 to 374.5 and latency rising from 1.2s to 11.3s. Users are encouraged to select a privacy setting that best aligns with their privacy needs and performance expectations, based on the provided configuration samples.

Smart device control. This paper omits low-level device control and focuses on the action plan generation stage, where the primary semantic privacy risk arises. In practice, major smart home ecosystems, such as Google Home[6], Amazon Alexa[7], and Mi Home[8], already expose official device control APIs, allowing legitimate third-party services to control local devices through well-defined interfaces. Therefore, *HomeLLaMA* is designed as a third-party service based on a local SLM, running entirely on local devices to perform natural language understanding and action planning without transmitting raw user commands to external LLM providers, which alleviates user privacy concerns significantly.

TABLE III
Privacy risks and costs of *PrivShield* in typical settings.

Metric \ Setting	$N = 0$	$N = 2$	$N = 4$	$N = 9$
Privacy Risk	100%	35.6%	22.3%	11.4%
Token Usage	33.2	91.6	167.1	374.5
Latency	1.2s	3.3s	6.1s	11.3s

IX. Conclusion

We present an on-device smart home assistant that strikes a balance between user privacy and performance. *HomeLLaMA* comprises three technical modules: *Local SLM Enhancement*, *Multi-party Interaction*, and *User Preference Learning*, enabling privacy-enhanced interactions that involve user-in-the-loop. We construct the *DevFinder* benchmark to evaluate the quality of action plans generated by smart home assistants. Comprehensive quantitative experiments and user studies demonstrate that *HomeLLaMA* delivers satisfactory plans with enhanced personalization and privacy protection.

References

[1] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.
[2] S. Solaimani, W. Keijzer-Broers, and H. Bouwman, "What we do–and don't–know about the smart home: an analysis of the smart home literature," *Indoor and Built Environment*, vol. 24, no. 3, pp. 370–383, 2015.
[3] X. Yu, Z. Zhou, L. Zhang, and X.-Y. Li, "Thumbup: Secure smartwatch controller for smart homes using simple hand gestures," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 865–878, 2022.

[6]https://developers.home.google.com/apis
[7]https://developer.amazon.com/en-US/docs/alexa/device-apis/smart-home-general-apis.html
[8]https://docs.api.xiaomi.com/en/cloud-ml/api/docs.html

[4] X. Huang, L. Shen, Z. Ma, and Y. Zheng, "Poster: Towards privacy-preserving and personalized smart homes via tailored small language models," in *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, 2025, pp. 1332–1334.

[5] J. Cui, J. Xiao, H. Zhong, J. Zhang, L. Wei, I. Bolodurina, and D. He, "Lh-ids: Lightweight hybrid intrusion detection system based on differential privacy in vanets," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 195–12 210, 2024.

[6] Z. Ma, L. Shen, X. Huang, and Y. Zheng, "Poster: Llmalware: An llm-powered robust and efficient android malware detection framework," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 4737–4739.

[7] E. King, H. Yu, S. Lee, and C. Julien, "Sasha: creative goal-oriented reasoning in smart homes with large language models," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 1, pp. 1–38, 2024.

[8] Z. Ramadan, M. F Farah, and L. El Essrawi, "From amazon. com to amazon. love: How alexa is redefining companionship and interdependence for people with special needs," *Psychology & Marketing*, vol. 38, no. 4, pp. 596–609, 2021.

[9] J. R. Bellegarda, "Spoken language understanding for natural interaction: The siri experience," *Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialog Systems into Practice*, pp. 3–14, 2013.

[10] D. Xu, W. Yin, H. Zhang, X. Jin, Y. Zhang, S. Wei, M. Xu, and X. Liu, "Edgellm: Fast on-device llm inference with speculative decoding," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 3256–3273, 2025.

[11] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[12] C. Liu and J. Zhao, "Enhancing stability and resource efficiency in llm training for edge-assisted mobile systems," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2025.

[13] Openai, "Gpt-4." [Online]. Available: https://chat.openai.com/chat/

[14] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *2012 IEEE international conference on communications (ICC)*. IEEE, 2012, pp. 917–922.

[15] T. Li, H. Wang, Q. Li, Y. Jiang, and Z. Yuan, "Cl-shield: A continuous learning system for protecting user privacy," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 3148–3162, 2025.

[16] W. Wang, Y. Wang, P. Duan, T. Liu, X. Tong, and Z. Cai, "A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5625–5642, 2022.

[17] H. Chi, Q. Zeng, X. Du, and J. Yu, "Cross-app interference threats in smart homes: Categorization, detection and handling," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 411–423.

[18] L. Shen, Q. Yang, K. Cui, Y. Zheng, X.-Y. Wei, J. Liu, and J. Han, "Fedconv: A learning-on-model paradigm for heterogeneous federated clients," in *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 2024, pp. 398–411.

[19] L. Shen and Y. Zheng, "Feddm: data and model heterogeneity-aware federated learning via dynamic weight sharing," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 975–976.

[20] L. Shen, Q. Yang, K. Cui, Y. Zheng, X.-Y. Wei, J. Liu, and J. Han, "Hierarchical and heterogeneous federated learning via a learning-on-model paradigm," *IEEE Transactions on Mobile Computing*, vol. 24, no. 11, pp. 12 103 – 12 120, 2025.

[21] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[22] D. Rivkin, F. Hogan, A. Feriani, A. Konar, A. Sigal, S. Liu, and G. Dudek, "Sage: Smart home agent with grounded execution," *arXiv preprint arXiv:2311.00772*, 2023.

[23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[24] E. King, H. Yu, S. Lee, and C. Julien, "Get ready for a party: Exploring smarter smart spaces with help from large language models," *arXiv preprint arXiv:2303.14143*, 2023.

[25] E. King, H. Yu, S. Vartak, J. Jacob, S. Lee, and C. Julien, "Thoughtful things: Building human-centric smart devices with small language models," *arXiv preprint arXiv:2405.03821*, 2024.

[26] S. Aheleroff, X. Xu, Y. Lu, M. Aristizabal, J. P. Velásquez, B. Joa, and Y. Valencia, "Iot-enabled smart appliances under industry 4.0: A case study," *Advanced engineering informatics*, vol. 43, p. 101043, 2020.

[27] M. Li, W. Gu, W. Chen, Y. He, Y. Wu, and Y. Zhang, "Smart home: architecture, technologies and systems," *Procedia computer science*, vol. 131, pp. 393–400, 2018.

[28] D. Marikyan, S. Papagiannidis, and E. Alamanos, "A systematic review of the smart home literature: A user perspective," *Technological Forecasting and Social Change*, vol. 138, pp. 139–154, 2019.

[29] R. D. Manu, S. Kumar, S. Snehashish, and K. Rekha, "Smart home automation using iot and deep learning," *International Research Journal of Engineering and Technology*, vol. 6, no. 4, pp. 1–4, 2019.

[30] A. S. Tulshan and S. N. Dhage, "Survey on virtual assistant: Google assistant, siri, cortana, alexa," in *Advances in Signal Processing and Intelligent Recognition Systems: 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers 4*. Springer, 2019, pp. 190–201.

[31] J. Li, Z. Li, G. Tyson, and G. Xie, "Characterising usage patterns and privacy risks of a home security camera service," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2344–2357, 2020.

[32] J. Xiao, Q. Zou, Q. Li, D. Zhao, K. Li, W. Tang, R. Zhou, and Y. Jiang, "User device interaction prediction via relational gated graph attention network and intent-aware encoder," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 1634–1642.

[33] J. Xiao, Q. Zou, Q. Li, D. Zhao, K. Li, Z. Weng, R. Li, and Y. Jiang, "I know your intent: Graph-enhanced intent-aware user device interaction prediction via contrastive learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 7, no. 3, pp. 1–28, 2023.

[34] H. Jeon, J. Kim, H. Yoon, J. Lee, and U. Kang, "Accurate action recommendation for smart home via two-level encoders and commonsense knowledge," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 832–841.

[35] L. Shen, Q. Yang, Y. Zheng, and M. Li, "Autoiot: Llm-driven automated natural language programming for aiot applications," in *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, 2025, pp. 468–482.

[36] L. Shen, Q. Yang, X. Huang, Z. Ma, and Y. Zheng, "Gpiot: Tailoring small language models for iot program synthesis and development," in *Proceedings of the 23rd ACM Conference on Embedded Networked Sensor Systems*, 2025, pp. 199–212.

[37] L. Shen and Y. Zheng, "Iotcoder: A copilot for iot application development," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 1647–1649.

[38] J. Rey-Jouanchicot, A. Bottaro, E. Campo, J.-L. Bouraoui, N. Vigouroux, and F. Vella, "Leveraging large language models for enhanced personalised user experience in smart homes," *arXiv preprint arXiv:2407.12024*, 2024.

[39] P. Sun, Z. Wang, L. Wu, Y. Feng, X. Pang, H. Qi, and Z. Wang, "Towards personalized privacy-preserving incentive for truth discovery in mobile crowdsensing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 352–365, 2020.

[40] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.

[41] H. Wu, W. Dai, W. Li, and Q. Yan, "Cape: Context-aware prompt perturbation mechanism with differential privacy," in *Forty-second International Conference on Machine Learning*, 2025.

[42] M. Tong, K. Chen, J. Zhang, Y. Qi, W. Zhang, N. Yu, T. Zhang, and Z. Zhang, "Inferdpt: Privacy-preserving inference for black-box large language models," *IEEE Transactions on Dependable and Secure Computing*, 2025.

[43] Q. Li, J. Wen, and H. Jin, "Governing open vocabulary data leaks using an edge llm through programming by example," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 4, pp. 1–31, 2024.

[44] P. M. Schwartz and D. J. Solove, "The pii problem: Privacy and a new concept of personally identifiable information," *NYUL rev.*, vol. 86, p. 1814, 2011.

[45] Z. Wang, Y. Sun, D. Liu, J. Hu, X. Pang, Y. Hu, and K. Ren, "Location privacy-aware task offloading in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 3, pp. 2269–2283, 2023.

[46] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *Pro-*

ceedings of the 10th international conference on Ubiquitous computing, 2008, pp. 202–211.

[47] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," *arXiv preprint arXiv:2011.01403*, 2020.

[48] H. Yu, J. Hua, and C. Julien, "Analysis of ifttt recipes to study how humans use internet-of-things (iot) devices," in *Proceedings of the 19th ACM conference on embedded networked sensor systems*, 2021, pp. 537–541.

[49] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," *arXiv preprint arXiv:2304.12244*, 2023.

[50] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in neural information processing systems*, vol. 36, pp. 10 088–10 115, 2023.

[51] Z. Han, C. Gao, J. Liu, S. Q. Zhang *et al.*, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *arXiv preprint arXiv:2403.14608*, 2024.

[52] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[53] U. Scherhag, C. Rathgeb, J. Merkle, and C. Busch, "Deep face representations for differential morphing attack detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3625–3639, 2020.

[54] Q. Li, J. Wen, and H. Jin, "Governing open vocabulary data leaks using an edge llm through programming by example," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 4, pp. 1–31, 2024.

[55] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[56] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *arXiv preprint arXiv:2401.08281*, 2024.

[57] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[58] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[59] S. Kim, S. Yun, H. Lee, M. Gubri, S. Yoon, and S. J. Oh, "Propile: Probing privacy leakage in large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 20 750–20 762, 2023.

[60] A. Frikha, N. Walha, K. K. Nakka, R. Mendes, X. Jiang, and X. Zhou, "Incognitext: Privacy-enhancing conditional text anonymization via llm-based private attribute randomization," *arXiv preprint arXiv:2407.02956*, 2024.

[61] Q. Li, J. Hong, C. Xie, J. Tan, R. Xin, J. Hou, X. Yin, Z. Wang, D. Hendrycks, Z. Wang *et al.*, "Llm-pbe: Assessing data privacy in large language models," *Proceedings of the VLDB Endowment*, vol. 17, no. 11, pp. 3201–3214, 2024.

[62] X. Hu, P.-Y. Chen, and T.-Y. Ho, "Radar: Robust ai-text detection via adversarial learning," *Advances in neural information processing systems*, vol. 36, pp. 15 077–15 095, 2023.

[63] o. Fabian Pedregosa, "Memory-Profiler: Monitor Memory usage of Python code." [Online]. Available: https://github.com/pythonprofilers/memory_profiler

[64] W. Huang, X. Zheng, X. Ma, H. Qin, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, and M. Magno, "An empirical study of llama3 quantization: From llms to mllms," *Visual Intelligence*, vol. 2, no. 1, p. 36, 2024.

[65] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[66] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[67] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[68] A. Joshi, S. Kale, S. Chandel, and D. K. Pal, "Likert scale: Explored and explained," *British journal of applied science & technology*, vol. 7, no. 4, pp. 396–403, 2015.

Xinyu Huang (Graduate Student Member, IEEE) is a PhD student in the Department of Computing, The Hong Kong Polytechnic University. Prior to that, he received his B.E. degree in Optoelectronic Information Science and Engineering from the University of Electronic Science and Technology of China. His research interests mainly lie in large foundation models with AIoT applications, embodied intelligence, security & privacy. He is a member of ACM.

Leming Shen (Graduate Student Member, IEEE) is a PhD student in the Department of Computing, The Hong Kong Polytechnic University. Prior to that, he received his B.E. degree in Software Engineering from Zhejiang University. His research interest mainly lies in Large Language Models, Mobile/Edge Computing, and IoT Applications. He is a member of ACM.

Zijing Ma (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in Computer Science and Technology from South China Agricultural University and Central South University in 2020 and 2023, respectively. He is currently pursuing a Ph.D. degree in the Department of Computing at The Hong Kong Polytechnic University. His research interests include mobile computing, mobile security, and large language models. He has published papers in journals and conferences, including IEEE Transactions on Mobile Computing and IEEE WCNC.

Yuanqing Zheng (Senior Member, IEEE) is an Associate Professor in the Department of Computing, the Hong Kong Polytechnic University. He received the B.S. degree in Electrical Engineering and the M.E. degree in Communication and Information System both from Beijing Normal University, Beijing, China. He received the Ph.D. degree in Computer Science from Nanyang Technological University, Singapore. His research interests include Internet of Things, Wireless Networking, Ubiquitous Computing, and Efficient AI. He is/was on the editorial board of Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (ACM IMWUT), ACM Transactions on Sensor Networks (ACM TOSN), Computer Networks, and IEEE Transactions on Wireless Communications (IEEE TWC). He is a member of ACM.