Portcullis: A Scalable and Verifiable Privacy Gateway for Third-Party LLM Inference

Jiangou Zhan[2 §], Wenhui Zhang[1 §], Zheng Zhang[1], Huanran Xue[1], Yao Zhang[1 †], Ye Wu[1 *]

[1] Bytedance Inc.
[2] Tsinghua University
[†] zhangyao.crypto@bytedance.com

Abstract

Businesses using third-party LLMs face privacy risks from exposed prompts. This paper presents Portcullis, a privacy-preserving gateway that safeguards sensitive data while supporting efficient and accurate LLM responses. Portcullis functions as a mediator, anonymizing sensitive data in prompts through parallel substitution, securely interacting with LLMs, and accurately reconstructing responses. It ensures that all data processing occurs within secure encrypted memory. The gateway is attested to ensure trustworthiness and protect user privacy. Portcullis is the first of its kind, offering a verifiable and scalable privacy gateway for third-party LLM inferences.

We assess Portcullis's efficiency as a confidential container platform, demonstrating that its startup time scales linearly, ensuring scalability. Additionally, we evaluate its runtime performance using the PII and Enron Email Dataset. For masking and unmasking workloads, Portcullis outperforms Hide-and-Seek by 96x speed up, while maintaining equal or better false positive and false negative rates compared to existing solutions. On the Enron dataset, Portcullis achieves notably higher accuracy, surpassing Hide-and-Seek by over 0.1 for GPT-4o mini.

Introduction

Large language models (LLMs) offer high-quality responses across a myriad of tasks and queries. Not all companies possess LLMs with robustness and superior performance, so they often use third-party LLMs like OpenAI's ChatGPT (Brown et al. 2020) to enhance their offerings. For instance, Apple leverages OpenAI APIs to deliver exceptional results for customers' prompt queries (OpenAI 2024; Apple 2024), thereby enhancing user experience and operational efficiency. However, the use of third-party LLMs introduces substantial privacy concerns. Sensitive information embedded in the prompts, such as personal identifiers (Macioce Jr 2018), financial data, and proprietary business information, may be inadvertently exposed to external entities. Given that third-party LLMs do not inherently eliminate sensitive information (Wang et al. 2023a), the potential leakage of such data poses a significant threat (Wu et al. 2025). Centralized entities operating these models could potentially exploit this vulnerability to orchestrate targeted attacks. Consequently, it is important to implement safeguards that ensure the confidentiality of input data while harnessing the functionalities of LLMs.

To address privacy issues in LLMs, previous studies have developed techniques such as fine-tuning pre-trained LLMs to process encrypted prompts, thus maintaining privacy without restructuring the underlying model (Mishra, Li, and Deo 2024). Additionally, methods like data obfuscation for TEE (Zhang et al. 2024; Tramer and Boneh 2018), implementing Differential Privacy (DP) (Du et al. 2023; Hou et al. 2024), Secure Multi-Party Computing (SMPC) (Hou et al. 2023), and hybrid cryptology methods (Lu et al. 2023) have been used to prevent data leakage. However, these methods require modifications to third-party LLMs, which are not possible for closed-source models like those from OpenAI (GPT-4o 2024). Modifications can also compromise the accuracy of their results. Additionally, they complicate deployment and increase operational overhead, leading to reduced Time To First Token (TTFT) and Time Per Output Token (TPOT). Apple Intelligence utilizes Private Cloud Compute (Apple 2024) to host LLMs, which enhances performance but may still pose risks of exposing sensitive information to third-party LLMs. Previous approaches (Chen et al. 2023; Tong et al. 2023) develops privacy-preserving prompt filters that function in the cloud to prevent the transmission of sensitive data to third-party LLMs. However, deploying these privacy filter methods on cloud platforms introduces inherent risks of data breaches, which could compromise data security (Huang, Shao, and Chang 2022; Huang et al. 2023).

We introduce Portcullis, a trustworthy gateway that intermediates between customers and third-party LLMs. To our knowledge, Portcullis is the first of its kind, capable of providing privacy-preserving prompts with accurate LLM responses in a verified fashion. Portcullis enhances privacy by obfuscating sensitive information in prompts using a chain of semantic-aware NERs and pattern matchers before sending them to third-party LLMs. It preserves the efficacy of LLM responses by securely restoring original meanings through a Redis Key-Value Store. It processes data within Intel TDX, where attestation verifies that data is handled

[*†] Corresponding author. § Co-first author, work done at ByteDance.

Scheme	Method	Practical	Accuracy	Security	Efficiency
DP-Forward (Du et al. 2023)	DP on Embedding	○	○	●	○
inferDPT (Tong et al. 2023)	DP on Text	●	○	◐	●
CipherGPT (Hou et al. 2023)	Multi Party Computation	○	○	●	○
BumbleBee (Lu et al. 2023)	Hybrid Cryptology	○	●	●	◐
Hide&Seek (Chen et al. 2023)	Mode-based	●	●	◐	○
GroupCover (Zhang et al. 2024)	TEE	○	●	●	◐
Slalom (Tramer and Boneh 2018)	TEE	○	●	●	●
Portcullis	Multi-level	●	●	◐	●

Table 1: Prior Works in Privacy-Preserving LLMs.

in encrypted memory. Additionally, application-level attestation ensures that the Portcullis code remains unaltered, adheres to the Execution Policy Actions, which prevents malicious behavior from Cloud Service Providers (CSPs). Evaluations confirm its effectiveness in filtering sensitive content while preserving response quality, with benchmarks highlighting its superior performance and scalability with minimal overhead. Contributions of Portcullis include:

- **Portcullis as a Privacy Gateway.** Portcullis sits as a gateway agent for interfacing with third-party LLMs. Portcullis detects and filters sensitive information in prompts and restores original privacy data in LLM responses to maintain meaning and clarity.
- **Portcullis is Secure.** Portcullis securely processes prompts and responses within encrypted memory, ensuring data integrity. It demonstrates a leak-proof operation by successfully passing the no-leak Bandit test.
- **Portcullis is Verifiable.** Portcullis integrates container attestation with tenant-defined execution policies, see Table. 2, and ensures trustworthiness and transparency through hardware-rooted attestation.
- **Portcullis is Scalable and Effective.** We demonstrate Portcullis's scalability with linearly scaling startup time and evaluate its runtime using the PII and Enron Email datasets. Portcullis achieves 96x speedup over Hide-and-Seek, with equal or better false positive/negative rates, and surpasses Hide-and-Seek by over 0.1 in accuracy on the Enron dataset for GPT-4o mini.

Background and Related Works

This section outlines the fundamental concepts of TEE and examines the implementation of privacy-preserving gateways in systems that serve LLMs.

Trusted Execution Environment

A Trusted Execution Environment (TEE) provides a secure enclave that protects data-in-use from threats, including those posed by malicious host kernels or hypervisors. Over the past two decades, TEE has been the subject of significant academic and industrial research, with developments across various hardware platforms like Intel SGX (McKeen et al. 2016; El-Hindi et al. 2022), Intel TDX (Cheng et al. 2023; Intel 2023), ARM Trustzone (Pinto and Santos 2019), ARM CCA (Zhang et al. 2023), AMD SEV (Kaplan, Powell, and Woller 2016), AMD SEV-SNP (Sev-Snp 2020), and RISC-V Keystone (Lee et al. 2020).

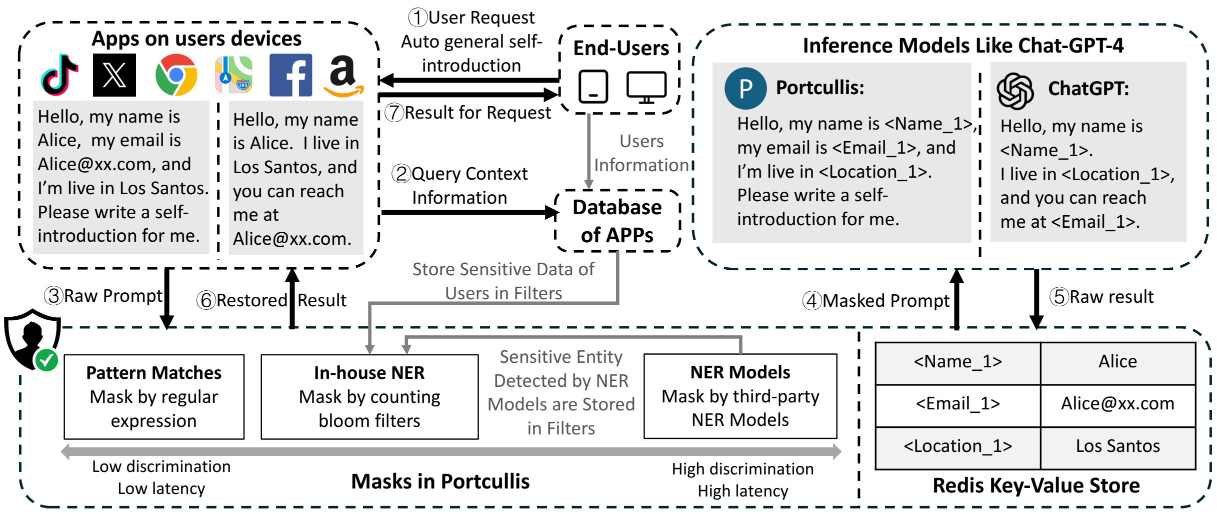
Portcullis leverages Intel Trusted Domain Extension (TDX) to offer a TEE-based virtual machine environment that ensures data confidentiality through encrypted memory and registers, integrity measurement, and remote attestation. Unlike Intel SGX, Intel TDX VMs do not necessitate the development of a library OS for application support, saving engineering efforts. Additionally, Intel TDX VMs can fully utilize all CPU and memory resources on a physical node, allowing for efficient management of large-memory workloads within secure memory, which reduces I/O operations and significantly enhances performance (Cheng et al. 2023; Intel 2023).

TDX utilizes Secure Arbitration Mode (SEAM) and Multi-key Total-Memory Encryption (MKTME) to ensure distinct VMs are isolated and encrypted with unique keys. SEAM mode provides a suite of instructions for secure interactions between the Trust Domain OS and the host/VMM, while MKTME manages memory encryption keys for enhanced security of the VMs. The attestation process within TDX includes generating a TDREPORT and a subsequent Quote through Intel SGX's Quoting Enclave (McKeen et al. 2016; El-Hindi et al. 2022) to verify the TD Guest's security and integrity. This attestation ensures that external parties can trust the VM's operational environment, enhancing the overall security framework for virtual environments.

Privacy-preserving Techniques for LLM

Methods to mitigate privacy issues in LLMs are classified into Privacy-preserving Models and Privacy-preserving Prompt Filters, as detailed in Table. 1.

Privacy-preserving Models. Privacy-preserving Models employ cryptographic technologies, such as Homomorphic Encryption (HE) (Chen et al. 2022b; Lu et al. 2023; Mishra et al. 2024), Differential Privacy (DP) (Du et al. 2023; Mai et al. 2023; Chen et al. 2022a; Hou et al. 2024), Secure Multi-Party Computing (SMPC) (Wang et al. 2022; Akimoto et al. 2023) and TEE-enhanced secure inference (Zhang et al. 2024), to secure prompt data. However, these Privacy-preserving Models demand extra computational resources and introduce increased complexity and overhead. For example, HE enables computations directly on encrypted data, producing precise results upon decryption. However, adapting LLM functions to support HE often requires substantial modifications. DP enhances privacy by



Apps on users devices
Hello, my name is Alice, my email is Alice@xx.com, and I'm live in Los Santos. Please write a self-introduction for me.
Hello, my name is Alice. I live in Los Santos, and you can reach me at Alice@xx.com.
①User Request Auto general self-introduction
⑦Result for Request
End-Users
Users Information
②Query Context Information
Database of APPs
Store Sensitive Data of Users in Filters
Inference Models Like Chat-GPT-4
Portcullis:
Hello, my name is <Name_1>, my email is <Email_1>, and I'm live in <Location_1>. Please write a self-introduction for me.
ChatGPT:
Hello, my name is <Name_1>. I live in <Location_1>, and you can reach me at <Email_1>.
③Raw Prompt
⑥Restored Result
④Masked Prompt
⑤Raw result
Pattern Matches Mask by regular expression
In-house NER Mask by counting bloom filters
Sensitive Entity Detected by NER Models are Stored in Filters
NER Models Mask by third-party NER Models
Low discrimination Low latency
Masks in Portcullis
High discrimination High latency
<Name_1> Alice
<Email_1> Alice@xx.com
<Location_1> Los Santos
Redis Key-Value Store

Figure 1: Portcullis employs pattern matching and machine learning techniques to detect sensitive information within datasets, subsequently substituting it with placeholders. To preserve the data's contextual integrity, it securely stores key-value pairs of sensitive information in an encrypted Redis, ensures that modified prompts maintain their relevance.

injecting noise into data or computations at various model layers to ensure outputs do not compromise individual data privacy, though this highly reduce result accuracy. TEE-based secure inference framework can achieve a good balance of efficiency and security without model modifications. However, they necessitate retuning third-party LLM serving systems, which is impractical.

Privacy-preserving Prompt Filters. Prompt Filters mitigate the risk of exposing sensitive information in third-party LLMs using pattern matching and parallel substitution. Pattern matching stage utilizes predefined patterns or regular expressions to detect common types of sensitive data, such as personally identifiable information (PII) (Schwartz and Solove 2011). Parallel data substitution stage aims to obfuscate sensitive information by replacing it with generalized or synthetic data, such as tokenization (Kulkarni and Cauvery 2021), pseudonymization (Deshpande 2021). Replacing personal identifiers with generic or synthetic data can effectively preserve privacy in straightforward scenarios, but this approach may struggle with context-sensitive data, leading to over-filtering or under-filtering, which could compromise LLM response quality.

Advanced Nature Language Processing (NLP) techniques such as NER (Mansouri, Affendey, and Mamat 2008), Contextual Analysis (Macioce Jr 2018), and Sentiment Analysis (Kulkarni and Cauvery 2021) improve the detection and obfuscation of sensitive data in prompts. These methods enable models to pre-filter sensitive information using enhanced contextual data before interfacing with LLMs (Tong et al. 2023; Chen et al. 2023; Shen et al. 2024). While these techniques demand significant memory and computational resources, making cloud-based processing ideal. However, the shared nature of cloud environments poses substantial trust and security challenges (Wang et al. 2023b).

Portcullis

As depicted in Figure. 1, Portcullis comprises both an infrastructure layer and an application layer. In its infrastructure layer, Portcullis leverages Intel TDX (Intel 2023) to generate attestation reports, ensuring that the integrity of the container group is aligned with the security policies from the start-up. It features a Portcullis agent restricted by execution policies to secure the container lifecycle, as illustrated in Figure. 2. Kata-shim initiates Portcullis instances and manages container operations within secure VMs, encompassing steps such as pulling images, starting pods, launching containers, and starting Portcullis's userspace code. Additionally, Portcullis's attestation mechanism rigorously confirms the security and integrity of these steps, enhancing trust and transparency. At the application layer, Portcullis integrates sophisticated components to strengthen privacy controls. Using advanced pattern matching and machine learning, the algorithm detects and manages sensitive information within the data. Recognized sensitive data is substituted with placeholders to maintain privacy. Importantly, Portcullis ensures the contextual integrity of the data is preserved, making sure that even after modifications, the prompts remain coherent and meaningful for processing by external LLMs. Additionally, Portcullis restores the original data from masked versions before delivering responses to users, ensuring that data recovery maintains the integrity of the original information.

Confidential Runtime and Transparency

Classical cloud infrastructure supporting large models has three primary attack surfaces. ① The first involves attackers exploiting vulnerabilities in the virtualization platform to escape from a virtual machine (VM), gaining control of the host machine and potentially compromising other VMs. For example, a 2019 vulnerability in QEMU-KVM (Tencent 2019) allowed attackers to escape during live VM migration and control the host machine. ② The second attack surface involves CSPs, who can manage container resources, such

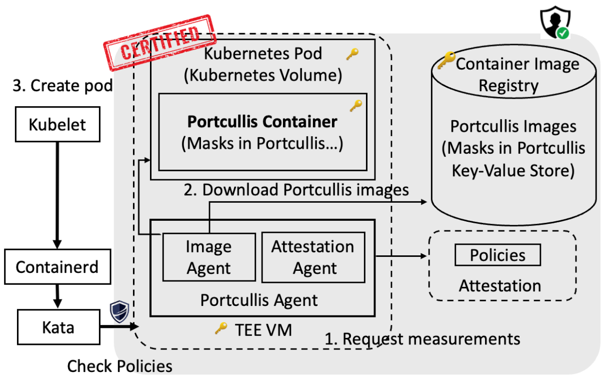


Figure 2: Portcullis Deployment. Untrusted kubelet triggers the kata-shim, where policies verification to ensure only trusted invocations are permitted.

Enforced Actions	Command Properties
Create container	ownerID, command, environment, working directory, mounts
Shutdown container	ownerID
Signal process	ownerID, signal, command
Execute process	ownerID, command, environment, working directory
Receive raw prompt	ownerID, command, raw prompt
Send masked prompt	ownerID, command, masked prompt
Receive raw response	ownerID, command, raw response
Send final response	ownerID, command, final response
Logging	ownerID

Table 2: Portcullis Execution Policy Actions.

as attaching volumes. By exploiting this control, CSPs can manipulate data on volumes, potentially replacing them with compromised versions, accessing sensitive information, or injecting malicious code. ③ The third attack surface pertains to the vulnerabilities of software running within containers, where flaws like SQL injection (Halfond et al. 2006) or XSS (Gupta and Gupta 2017) in the application code can create leak points for sensitive data. Securing the containers alone is insufficient; comprehensive security measures are needed to address both infrastructure and application vulnerabilities.

Portcullis eliminates the first attack surface by enhancing data confidentiality through the use of CoCo (Community 2024). This approach processes all data within encrypted memory and ensures integrity via attestation. Portcullis permits only verified containers to execute, continuously monitoring for any tampering that could affect deployment or compromise the integrity of remote attestations. Figure. 2 illustrates how Portcullis secures container root file systems against CSP threats. Similar to CoCo, it implements dm-verity for integrity protection and mounts container image layers in a read-only, encrypted format. Additionally, Portcullis incorporates block-level encryption and integrity checks, confining sensitive data to plaintext exclusively within the secure memory of the VM. This approach safeguards both the writable layer and remote blob storage.

To enhance security and mitigate malicious requests from kata-shim to the Portcullis agent, as well as to prevent unexpected behavior from the CSPs, Portcullis incorporates a tenant-defined execution policy. This policy specifies allowable actions for the Portcullis agent, which include mounting/unmounting devices, processing user inputs, interfacing with third-party LLMs, and managing responses. These permissible operations are detailed in Table. 2. Portcullis performs comprehensive mediation and attestation for all actions within the CoCo instance, reporting them in a verifiable manner. The attestation mechanism cross-checks each action against the tenant-defined execution policy and sends alerts to tenants if an action deviates from the approved whitelist. Tenants can then decide whether to abort the deployment or disregard the potentially malicious behaviors initiated by CSPs. Moreover, during the initialization of Portcullis, this policy is embedded into the attestation report as an immutable field, ensuring the policy's integrity and verifiability throughout the lifecycle of the application.

Portcullis facilitates the detection of sensitive information, data obfuscation, and recovery processes within the protective framework of the previously mentioned methods. Despite these safeguards, the software remains vulnerable to common threats, such as SQL injection or XSS, which could lead to data leakage. To combat these vulnerabilities, Portcullis integrates Bandit (Peng, Liu, and Han 2019), a static analysis tool tailored for Python codebases. Bandit constructs an Abstract Syntax Tree (AST) and leverages a comprehensive array of security plugins to examine the AST thoroughly. This method effectively identifies and addresses 78 different security issues, including SQL and shell injections, and hardcoded passwords. Portcullis has been adapted to ensure compatibility with Bandit's standards, resulting in zero reported issues.

Sensitive Information Detection

Portcullis integrates pattern matching and machine learning to identify sensitive information within input prompts.

Taxonomy of Privacy Data. The prevailing taxonomy for sensitive data primarily targets Personally Identifiable Information (PII), yet it fails to cover other critical sensitive information such as proprietary corporate details or political data. This might include specifics like corporate management protocols or annual budget. Moreover, the sensitivity of information depends on the context in which it is situated. For instance, the term 'Donald Trump' could denote a Las Vegas building rather than the person, altering its sensitivity based on the reference. While the name might be sensitive when referring to the person, it becomes non-sensitive when associated with a building. Moreover, even when referencing the individual, the context can make the name non-sensitive. For example, in scenarios like tallying election votes, the name 'Donald Trump' alongside vote counts is publicly disclosed, making it non-sensitive.

Portcullis enhances the precision of context-based privacy by employing a sensitive data-sensitive database tagged by users to discern pertinent content within specific contexts, thus minimizing false positives and negatives. For example, when processing news related to 'Donald Trump', the system evaluates whether the context encompasses sensitive information requiring protection. In a non-sensitive context, if Trump is not a user of the platform, articles concerning his public speeches or policies are deemed non-sensitive due to their public nature. Conversely, in a sensitive context, if Trump were a platform user, private details such as personal communications or undisclosed financial information would be classified as sensitive; otherwise, such information remains non-sensitive.

Pattern Matching. Portcullis employs regular expressions, in-house NER, Presidio (Mendels et al. 2018) and BERT-NER (Liu et al. 2021) to effectively identify and classify common types of sensitive information. To accelerate processing, Portcullis utilizes a caching layer that quickly identifies and handles previously encountered sensitive named entities, overcoming the inefficiencies of token-by-token pattern matching.

Portcullis follows a cascading layered approach for pattern matching, optimizing both efficiency and accuracy. Moreover, it allows for flexible reordering of these layers to better adapt to varying application environments. Initially, Portcullis leverages regular pattern recognition or detecting sensitive data such as names, addresses, and financial details. To manage more nuanced and context-dependent sensitive data, Portcullis integrates advanced NLP engines from Presidio (Mendels et al. 2018). This approach involves configuring custom models and utilizing them within an *AnalyzerEngine* for enhanced detection capabilities. Portcullis creates an *NlpEngine* using the *NlpEngineProvider* class and pass it to the *AnalyzerEngine*. Additionally, Portcullis defines which entities should be ignored or how to map model labels to Presidio entity types, and adjust confidence scores based on accuracy requirements. Finally, Portcullis trains an in-house NER for contextually defined priavcy data. Portcullis queries in-house database, train a model based on context-dependent sensitive data taxonomy, and identifies sensitive name entities.

Portcullis mitigates the performance overhead of running computationally intensive NER engines during runtime by pre-processing sensitive data recognition in the background. To prevent redundant queries and execution of NER engines in real-time, Portcullis incorporates a bloom filter, serving as an efficient caching layer for swift pattern matching. The bloom filter employed by Portcullis is designed as a counting bloom filter, which not only facilitates dynamic adjustments, such as adding or removing entries to keep pace with changes in real-world scenarios, but also records sensitive data patterns, ensuring the swift identification of sensitive entities.

Data Obfuscation and Recovery

Once sensitive information is detected, the next step is to obfuscate it to ensure privacy while preserving the contextual integrity of the prompt.

Sensitive Data Obfuscation. Our method for replacing sensitive information incorporates several strategies aimed at maintaining data privacy without compromising the utility of the data for analytical purposes. Tokenization involves substituting identifiable data elements with non-specific placeholders, such as [NAME] or [ADDRESS], ensuring the structure of the data remains intact for processing while obfuscating the identifiable details. Pseudonymization replaces real data points with fictitious, yet plausible alternatives, allowing for data utility in scenarios where the actual data's context is necessary. Synthetic Data Generation creates entirely new datasets that mimic the statistical properties of the original data but do not contain any actual sensitive information. For example, in *Bob and Alice are going to New York, and their email is bob@gmail.com*, the names 'Bob' and 'Alice' in a sentence might be replaced with [NAME_1] and [NAME_2], and geographical and contact details transformed to [LOCATION_1] and [EMAIL_1], respectively. The masked result is *[NAME_1] and [NAME_2] are going to [LOCATION_1] and their email is [EMAIL_1].* These techniques are designed to strike a balance between protecting individual privacy and allowing for meaningful data analysis.

Sensitive Data Recovery. Portcullis processes obfuscated prompts through LLMs to generate responses that adhere to the same obfuscation rules used during the initial data masking phase. This ensures that any sensitive information remains protected throughout the interaction. Post-inference, Portcullis retrieves the original data associations from a Redis cluster (Carlson 2013), which sits in encrypted memory and securely stores mappings of obfuscated tokens to their original data. During data recovery, Portcullis utilizes these mappings to translate the obfuscated tokens back to their original sensitive data. This reversal of the obfuscation process enables the restoration of the response to its original, meaningful form, ensuring that users receive accurate and contextually coherent information without compromising privacy.

Discussion on Security Guarantees

Portcullis processes all data within encrypted memory, and perform attestation to safeguard the confidentiality and integrity of Portcullis itself. There are some orthoganal security concerns which could be handled by Portcullis as well.

While Portcullis adheres to Bandit's standards without any reported vulnerabilities and protect the tenant from itself, it is important to recognize that no tool can entirely eliminate the risk of data breaches. The effectiveness of Bandit (Peng, Liu, and Han 2019), which we utilize alongside other static and dynamic analysis tools, is contingent upon the range of its plugins and how well they align with the specific applications of Portcullis. While this issue is tangential to our main objectives, Portcullis is designed to be thoroughly tested by static and dynamic analysis tools prior to deployment.

Secondly, while all sensitive operations conducted within a secure enclave protect data from external exposure and unauthorized access, Portcullis is still vulnerable to differential privacy attacks. This type of attack exploits the inherent trade-offs in differential privacy techniques, which balance privacy protection with the utility of the data. To address this orthogonal issue, Portcullis framework allows integration differential privacy protection engines into Portcullis's privacy masking chain. This addition would enhance the system's ability to safeguard against such sophisticated threats,

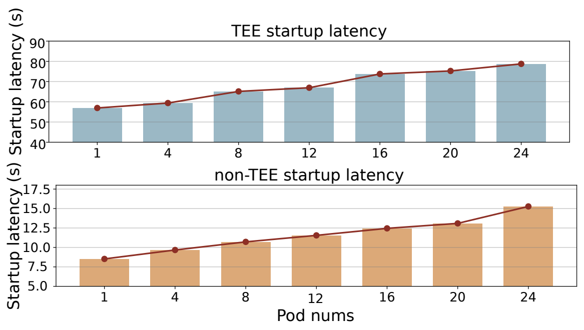


Figure 3: Startup latency trend while pods scaling. The more linear the latency increase as the scale grows, the better the scalability.

further strengthening the privacy and security of user data. Portcullis leverages Intel TDX to address side-channel vulnerabilities, with firmware updates resolving recent TEE attacks like TDXDown (Wilke, Sieck, and Eisenbarth 2024). Further defenses, such as Dove (Lee et al. 2021) and QuanShield (Cui et al. 2023), can enhance Portcullis's security posture.

Lastly, the sensitivity of entities identified by Portcullis can vary with context. For example, *Donald Trump* may not be sensitive when referring to a building in Las Vegas. To improve accuracy, Portcullis allows users to provide additional context via an in-house NER plugin, ensuring more precise results.

In summary, Portcullis enables secure processing within encrypted memory and robust attestation, offering a leak-proof privacy gateway that filters and restores sensitive content in LLM interactions, demonstrating trustworthiness in handling privacy data effectively.

Evaluation and Analysis

System Setup. Our experiments are conducted on a physical node equipped with the fourth-generation Intel® Xeon® Scalable Processor (Sapphire Rapids), with a base frequency of 2.6GHz and an all-core turbo frequency of 3.1GHz. We install bare-metal machine with Kubernetes v1.28.2, QEMU 8.1.4, Debian 12 and Linux version 5.15.120 with a back-ported TDX patch from Linux version 6.3.0. All tests are conducted on the CPUs of NUMA node0, with each container instance allocated 32 GB of RAM, and 16 vCPU cores. Container instances operate on Debian 12 as the guest image. Confidential containers employ a guest kernel from Linux version 5.15.120, enhanced with a TDX patch back-ported from Linux version 6.3.0. In contrast, regular containers also use Debian 12 but with the unmodified Linux version 5.15.120.

Baseline. We apply four different masking techniques in Portcullis: Pattern Matches, In-house NER, Presidio (Mendels et al. 2018), and Bert-NER (Liu et al. 2021). We conduct a comprehensive evaluation to demonstrate the efficiency of system, effectiveness in identifying privacy entities, and the response accuracy of masked prompts. For the efficiency evaluation, we combine these four masks to ensure fairness. In the evaluation of effectiveness and accuracy, each of the four masks is tested individually to provide a more detailed performance analysis. Additionally, we compare the performance of Presidio with other prompt privacy protection schemes, including Hide and Seek (Chen et al. 2023) and Infer-DPT (Tong et al. 2023). These two schemes treat LLMs as black-box models.

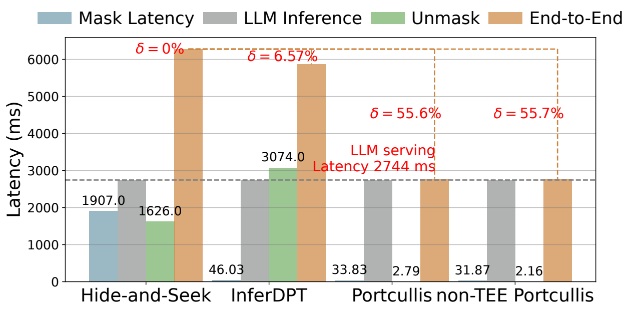


Figure 4: Latency comparison across different schemes. The lower the better.

	Dataset	False Negative	False Positive	Full Match
Hide-and-Seek	PII	0.2176	$54.78e^{-4}$	0.663
	Enron	0.4461	0.0998	0.009
Portcullis RegExp.	PII	0.7861	$2.12e^{-4}$	0.136
	Enron	0.3105	0.0545	0.142
Portcullis In-house NER	PII	0.6346	$34.72e^{-4}$	0.255
	Enron	0.0089	$1.78e^{-6}$	0.987
Portcullis Presidio	PII	0.2655	$5.71e^{-4}$	0.650
	Enron	0.0427	0.0195	0.066
Portcullis BERT-NER	PII	0.1103	$19.64e^{-4}$	0.824
	Enron	0.0220	0.0208	0.054

Table 3: Effectiveness by Hamming Distance. False negatives and positives are lower the better. The higher the full match the better.

Datasets and Models. The evaluation of Portcullis is performed using two datasets: the PII dataset (ai4Privacy 2023) and the Enron Email dataset (Mazeika et al. 2024; Shetty and Adibi 2004), which encompass various types of sensitive information. The experiments utilize OpenAI's GPT-4o mini (OpenAI 2024; GPT-4o 2024) and Mistral-7B (AI 2024) with context window of 32K to assess Portcullis's effectiveness in real-world scenarios.

Boot-time and Runtime Efficiency

Startup Latency. We assess the scalability of Portcullis by comparing the startup latency when concurrently launching multiple instances, as shown in Figure. 3. As the number of pods increases from 1 to 24, the service startup latency grows linearly, demonstrating robust scalability. Additionally, we deployed a comparable service in a non-TEE environment as a baseline to assess the overhead introduced by Portcullis's verifiable feature. The results indicate that the introduction of the Confidential Container during pod scaling did not negatively affect startup latency, underscoring the excellent scalability of our solution.

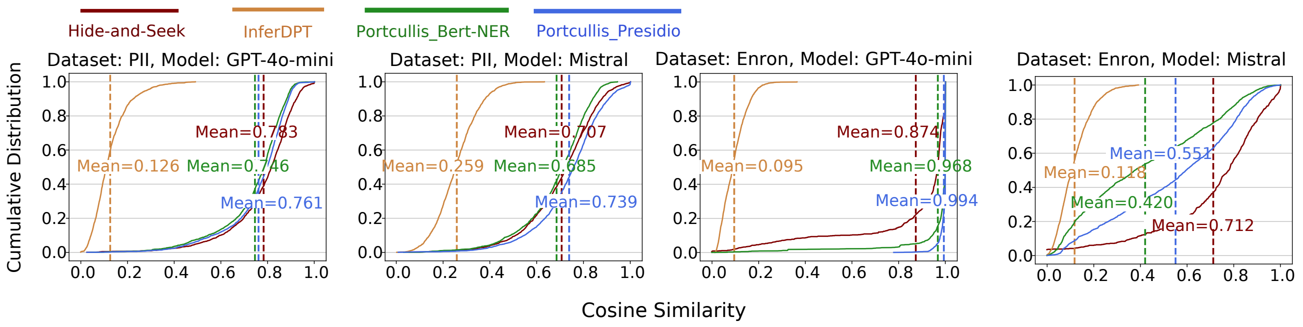


Figure 5: Accuracy by Cosine Similarity. Higher the better.

Runtime Latency. We evaluate Portcullis with LLaMA-2 7B (Touvron et al. 2023) using the vLLM framework (Kwon et al. 2023), measuring runtime latency from data token reception to final output delivery. This process includes data masking, processing through the LLM, unmasking, and final output stages. To ensure accuracy, we conducted tests on end-to-end service latency across 100 requests to avoid performance jitters. Latency is reported at each processing stage, representing the combined time required to process the initial and all subsequent tokens. The results are shown in Figure. 4. Firstly, the overhead introduced by Portcullis is negligible, amounting to only 1.33% compared to third-party LLM inference. Secondly, Portcullis outperforms both Hide-and-Seek and InferDPT in mask and unmask latency. Lastly, compared to the non-TEE solution, the mask performance degradation is within 10%, and the unmask performance remains similar, with a latency of less than 1 ms.

Effectiveness and Accuracy

Effectiveness of Identifying Privacy Entities. We assess the effectiveness of Portcullis in masking sensitive information by evaluating its success rate on the PII_masking-43K dataset. We benchmark these results against the sensitive information labels provided within the PII_masking-43K dataset to quantitatively measure performance. The evaluation demonstrates that as the masks in Portcullis progress from simple to complex, both the false negative rate and full match hit rate decrease. With the most complex mask, Portcullis achieves a 25.8% superior performance compared to Hide-and-Seek, as shown in Table. 3, with a false negative rate of 0.11, a false positive rate of 0.02, and a full match hit rate of 0.82.

To evaluate the effectiveness of Portcullis in scenarios where privacy information is defined within a database, we use the Enron email dataset, which includes user names, email addresses, and the content of emails. The results are shown in Table. 3. Our in-house NER mask demonstrates excellent performance compared to other schemes, particularly in terms of false positive rate, which is approximately 1,000 times lower than that of other methods.

Responses Accuracy. To assess response accuracy, we compute cosine similarity through the following procedure: First, both the raw and filtered prompts are submitted to the same LLM, yielding responses denoted as res_{raw} and res_{fil}. Next, we calculate the Frequency-Inverse Document Frequency (TF-IDF) scores for each word in these responses, transforming them into numerical vectors vec_{raw} and vec_{fil}. Finally, the cosine similarity between these vectors as shown in Formular. 1.

$$CosineSimilarity := \frac{\overrightarrow{vec_{raw}} \cdot \overrightarrow{vec_{fil}}}{\|\overrightarrow{vec_{raw}}\| \|\overrightarrow{vec_{fil}}\|}, \tag{1}$$

Accuracy of Portcullis and others two schemes are evaluated using the four combination of datasets and models, as shown in Figure. 5. For the PII dataset, Portcullis achieves a similar average accuracy to Hide-and-Seek across both models, with approximately 0.75 in GPT-4o mini and 0.7 in Mistral. This result indicates that Portcullis requires significantly less time to achieve comparable accuracy in responses. For the Enron dataset, which consists of easier tasks but more diversified data, GPT-4o mini achieves significantly higher accuracy with Portcullis, exceeding Hide-and-Seek by more than 0.1. However, the performance of Portcullis in Mistral is less satisfactory, possibly due to Mistral's limited ability to comprehend complex data, resulting in more random outcomes. The response accuracy of InferDPT is very low, as the added noise in the prompts prevents the LLMs from fully understanding the specific meaning of the queries.

Conclusion

This paper presents Portcullis, a scalable and verifiable privacy gateway designed to enhance the privacy of prompts to LLMs while minimally compromising response quality. We develop a credible and secure framework within a TEE enclave and implement various masks in Portcullis, offering users ready-to-use services. Our results demonstrate that Portcullis effectively reduces the exposure of sensitive information while maintaining LLM response quality with minimal degradation, achieving an accuracy rate above 0.7. The computational cost remains within acceptable bounds, with a linear-growing startup latency when launching multiple instances concurrently and a response latency of approximately 3 seconds, making it both scalable and efficient for real-time applications.

Acknowledgments

We thank OpenAI for their research grant support on tokens and extend our gratitude to Ron Perez and Ken Lu for their valuable feedback on Intel TDX.

References

AI, M. 2024. Introducing Mistral. Available at https://mistral.ai/, Accessed: June 2024.

ai4Privacy. 2023. AI4Privacy PII-Masking Dataset: Safeguarding Personal Data in AI Interactions.

Akimoto, Y.; Fukuchi, K.; Akimoto, Y.; and Sakuma, J. 2023. Privformer: Privacy-preserving transformer with mpc. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 392–410. IEEE.

Apple. 2024. Introducing Apple Intelligence for iPhone, iPad, and Mac. Available at https://www.apple.com/newsroom/2024/06/introducing-apple-intelligence-for-iphone-ipad-and-mac/, Accessed: June 2024.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Carlson, J. 2013. *Redis in action*. Simon and Schuster.

Chen, H.; Mo, F.; Wang, Y.; Chen, C.; Nie, J.-Y.; Wang, C.; and Cui, J. 2022a. A customized text sanitization mechanism with differential privacy. *arXiv preprint arXiv:2207.01193*.

Chen, T.; Bao, H.; Huang, S.; Dong, L.; Jiao, B.; Jiang, D.; Zhou, H.; Li, J.; and Wei, F. 2022b. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216*.

Chen, Y.; Li, T.; Liu, H.; and Yu, Y. 2023. Hide and seek (has): A lightweight framework for prompt privacy protection. *arXiv preprint arXiv:2309.03057*.

Cheng, P.-C.; Ozga, W.; Valdez, E.; Ahmed, S.; Gu, Z.; Jamjoom, H.; Franke, H.; and Bottomley, J. 2023. Intel TDX Demystified: A Top-Down Approach. *arXiv preprint arXiv:2303.15540*.

Community, C. C. 2024. Confidential Containers.

Cui, S.; Li, H.; Li, Y.; Zhang, Z.; Vilanova, L.; and Pietzuch, P. 2023. QuanShield: Protecting against Side-Channels Attacks using Self-Destructing Enclaves. *arXiv preprint arXiv:2312.11796*.

Deshpande, A. 2021. Sypse: Privacy-first Data Management through Pseudonymization and Partitioning. In *CIDR*.

Du, M.; Yue, X.; Chow, S. S.; Wang, T.; Huang, C.; and Sun, H. 2023. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2665–2679.

El-Hindi, M.; Ziegler, T.; Heinrich, M.; Lutsch, A.; Zhao, Z.; and Binnig, C. 2022. Benchmarking the second generation of intel sgx hardware. In *Proceedings of the 18th International Workshop on Data Management on New Hardware*, 1–8.

GPT-4o, O. 2024. Hello GPT-4o.

Gupta, S.; and Gupta, B. B. 2017. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8: 512–530.

Halfond, W. G.; Viegas, J.; Orso, A.; et al. 2006. A Classification of SQL Injection Attacks and Countermeasures. In *ISSSE*.

Hou, C.; Shrivastava, A.; Zhan, H.; Conway, R.; Le, T.; Sagar, A.; Fanti, G.; and Lazar, D. 2024. PrE-Text: Training Language Models on Private Federated Data in the Age of LLMs. *arXiv preprint arXiv:2406.02958*.

Hou, X.; Liu, J.; Li, J.; Li, Y.; Lu, W.-j.; Hong, C.; and Ren, K. 2023. Ciphergpt: Secure two-party gpt inference. *Cryptology ePrint Archive*.

Huang, J.; Shao, H.; and Chang, K. C.-C. 2022. Are large pre-trained language models leaking your personal information? *arXiv preprint arXiv:2205.12628*.

Huang, Y.; Gupta, S.; Xia, M.; Li, K.; and Chen, D. 2023. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*.

Intel. 2023. Intel TDX Connect Architecture Specification.

Kaplan, D.; Powell, J.; and Woller, T. 2016. AMD memory encryption.

Kulkarni, P.; and Cauvery, N. 2021. Personally identifiable information (pii) detection in the unstructured large text corpus using natural language processing and unsupervised learning technique. *International Journal of Advanced Computer Science and Applications*, 12(9).

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Lee, D.; Kohlbrenner, D.; Shinde, S.; Asanović, K.; and Song, D. 2020. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, 1–16.

Lee, H. B.; Jois, T. M.; Fletcher, C. W.; and Gunter, C. A. 2021. DOVE: A data-oblivious virtual environment. *arXiv preprint arXiv:2102.05195*.

Liu, Z.; Jiang, F.; Hu, Y.; Shi, C.; and Fung, P. 2021. NER-BERT: a pre-trained model for low-resource entity tagging. *arXiv preprint arXiv:2112.00405*.

Lu, W.-j.; Huang, Z.; Gu, Z.; Li, J.; Liu, J.; Ren, K.; Hong, C.; Wei, T.; and Chen, W. 2023. Bumblebee: Secure two-party inference framework for large transformers. *Cryptology ePrint Archive*.

Macioce Jr, D. L. 2018. PII in context: video privacy and a factor-based test for assessing personal information. *Pepp. L. Rev.*, 45: 331.

Mai, P.; Yan, R.; Huang, Z.; Yang, Y.; and Pang, Y. 2023. Split-and-Denoise: Protect large language model inference with local differential privacy. *arXiv preprint arXiv:2310.09130*.

Mansouri, A.; Affendey, L. S.; and Mamat, A. 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2): 339–344.

Mazeika, M.; Phan, L.; Yin, X.; Zou, A.; Wang, Z.; Mu, N.; Sakhaee, E.; Li, N.; Basart, S.; Li, B.; et al. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

McKeen, F.; Alexandrovich, I.; Anati, I.; Caspi, D.; Johnson, S.; Leslie-Hurd, R.; and Rozas, C. 2016. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, 1–9.

Mendels, O.; Peled, C.; Vaisman Levy, N.; Rosenthal, T.; Lahiani, L.; et al. 2018. Microsoft Presidio: Context aware, pluggable and customizable PII anonymization service for text and images.

Mishra, A.; Asai, A.; Balachandran, V.; Wang, Y.; Neubig, G.; Tsvetkov, Y.; and Hajishirzi, H. 2024. Fine-grained hallucination detection and editing for language models. *arXiv preprint arXiv:2401.06855*.

Mishra, A.; Li, M.; and Deo, S. 2024. SentinelLMs: Encrypted Input Adaptation and Fine-tuning of Language Models for Private and Secure Inference. In *AAAI*.

OpenAI. 2024. OpenAI and Apple Announce Partnership to Integrate ChatGPT into Apple Experiences. Available at https://openai.com/index/openai-and-apple-announce-partnership/, Accessed: June 2024.

Peng, S.; Liu, P.; and Han, J. 2019. A Python security analysis framework in integrity verification and vulnerability detection. *Wuhan University Journal of Natural Sciences*, 24(2): 141–148.

Pinto, S.; and Santos, N. 2019. Demystifying arm trustzone: A comprehensive survey. *ACM computing surveys (CSUR)*, 51(6): 1–36.

Schwartz, P. M.; and Solove, D. J. 2011. The PII problem: Privacy and a new concept of personally identifiable information. *NYUL rev.*, 86: 1814.

Sev-Snp, A. 2020. Strengthening VM isolation with integrity protection and more. *White Paper, January*, 53: 1450–1465.

Shen, Z.; Xi, Z.; He, Y.; Tong, W.; Hua, J.; and Zhong, S. 2024. The Fire Thief Is Also the Keeper: Balancing Usability and Privacy in Prompts. *arXiv preprint arXiv:2406.14318*.

Shetty, J.; and Adibi, J. 2004. The Enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California*, 4(1): 120–128.

Tencent, B. T. 2019. QEMU-KVM Leaking. https://cloud.tencent.com/developer/article/1520414. Accessed: 2024-12-18.

Tong, M.; Chen, K.; Qi, Y.; Zhang, J.; Zhang, W.; and Yu, N. 2023. InferDPT: Privacy-Preserving Inference for Black-box Large Language Model. *arXiv preprint arXiv:2310.12214*.

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

Tramer, F.; and Boneh, D. 2018. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. *arXiv preprint arXiv:1806.03287*.

Wang, B.; Chen, W.; Pei, H.; Xie, C.; Kang, M.; Zhang, C.; Xu, C.; Xiong, Z.; Dutta, R.; Schaeffer, R.; et al. 2023a. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In *NeurIPS*.

Wang, Y.; Lin, Y.; Zeng, X.; and Zhang, G. 2023b. Privatelora for efficient privacy preserving llm. *arXiv preprint arXiv:2311.14030*.

Wang, Y.; Suh, G. E.; Xiong, W.; Lefaudeux, B.; Knott, B.; Annavaram, M.; and Lee, H.-H. S. 2022. Characterization of mpc-based private inference for transformer-based models. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 187–197. IEEE.

Wilke, L.; Sieck, F.; and Eisenbarth, T. 2024. TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 79–93.

Wu, G.; Zhang, Z.; Zhang, Y.; Wang, W.; Niu, J.; Wu, Y.; and Zhang, Y. 2025. I Know What You Asked: Prompt Leakage via KV-Cache Sharing in Multi-Tenant LLM Serving. In *Proceedings of the 2025 Network and Distributed System Security (NDSS) Symposium*. San Diego, CA, USA. ISBN 979-8-9894372-8-3.

Zhang, Y.; Hu, Y.; Ning, Z.; Zhang, F.; Luo, X.; Huang, H.; Yan, S.; and He, Z. 2023. SHELTER: Extending Arm CCA with Isolation in User Space. In *32nd USENIX Security Symposium (USENIX Security'23)*.

Zhang, Z.; Wang, N.; Zhang, Z.; Zhang, Y.; Zhang, T.; Liu, J.; and Wu, Y. 2024. GroupCover: A Secure, Efficient and Scalable Inference Framework for On-device Model Protection based on TEEs. In *Forty-first International Conference on Machine Learning*.