EncChain: Enhancing Large Language Model Applications with Advanced Privacy Preservation Techniques

Zhe Fu
Alibaba Cloud
jeff.fz@alibaba-inc.com

Mo Sha
Alibaba Cloud
shamo.sm@alibaba-inc.com

Yiran Li
Alibaba Cloud
yiranli.lyr@alibaba-inc.com

Huorong Li
Alibaba Cloud
huorong.lhr@alibaba-inc.com

Yubing Ma
Alibaba Cloud
yubing.myb@alibaba-inc.com

Sheng Wang
Alibaba Cloud
sh.wang@alibaba-inc.com

Feifei Li
Alibaba Cloud
lifeifei@alibaba-inc.com

ABSTRACT

In response to escalating concerns about data privacy in the Large Language Model (LLM) domain, we demonstrate **EncChain**, a pioneering solution designed to bolster data security in LLM applications. **EncChain** presents an all-encompassing approach to data protection, encrypting both the knowledge bases and user interactions. It empowers confidential computing and implements stringent access controls, offering a significant leap in securing LLM usage. Designed as an accessible Python package, **EncChain** ensures straightforward integration into existing systems, bolstered by its operation within secure environments and the utilization of remote attestation technologies to verify its security measures. The effectiveness of **EncChain** in fortifying data privacy and security in LLM technologies underscores its importance, positioning it as a critical advancement for the secure and private utilization of LLMs.

PVLDB Reference Format:
Zhe Fu, Mo Sha, Yiran Li, Huorong Li, Yubing Ma, Sheng Wang, and Feifei Li. **EncChain**: Enhancing Large Language Model Applications with Advanced Privacy Preservation Techniques. PVLDB, 17(12): 4413 - 4416, 2024.
doi:10.14778/3685800.3685888

1 INTRODUCTION

Since late 2022, interest in Large Language Models (LLMs) [1] has surged dramatically. ChatGPT, for instance, amassed over 100 million active users within just two months of its launch, representing an unprecedented technological uptake. The profound capabilities of LLMs across diverse domains have catalyzed their widespread adoption, integration efforts in various use cases, and demonstrated substantial benefits in augmenting productivity and efficiency.

However, the rapid advancement of LLMs has highlighted significant data security and privacy issues. These concerns are not merely theoretical. In March 2023, the Italian Data Protection Authority banned ChatGPT due to privacy concerns. In April, Samsung was accused of leaking sensitive semiconductor data to ChatGPT in three incidents over 20 days. By November, Microsoft prohibited employees from using ChatGPT at work, blocking related AI tools on company devices. These instances indicate a shift from initial enthusiasm to a more measured approach, recognizing the pronounced issues with LLMs in practical applications.

The aggregation of extensive knowledge bases and user queries, often containing sensitive data, introduces substantial security vulnerabilities when processed by LLMs. Typical LLM applications, such as third-party tailored domain-specific APIs, require significant computational resources and specialized hardware, often favoring cloud deployments. This setup introduces various security threats, including data exposure due to negligence or malicious service providers, multi-tenant architecture risks, and the potential misuse of sensitive user data for model refinement. The lack of theoretical tools to mitigate the risk of LLMs inadvertently revealing sensitive content further complicates the issue. As technological applications deepen, data security emerges as a pivotal constraint to the advancement of LLM technologies.

In this paper, we demonstrate the proposed **EncChain**—a novel privacy preservation solution tailored for LLM applications, underpinned by confidential data handling practices. The strategic application of **EncChain** significantly enhances data security measures within LLM frameworks, diminishing the likelihood of unauthorized data access and exploitation. More specifically, **EncChain** exhibits the following key attributes:

- **Encrypted Knowledge Base and User Interactions**: All knowledge base and interaction records are encrypted using distinct keys before leaving the secure perimeter, which ensures that information remains perpetually in ciphered form, thereby precluding access to its unencrypted counterpart, even for application architects.
- **Confidential Data Computing Capability**: **EncChain** provides a suite of core functionalities, including confidential knowledge base loading, confidential similarity search, confidential prompt generation, and confidential large model inference. These capabilities enable developers to handle and process encrypted data without accessing plaintext, meeting the requirements for constructing business logic while protecting data privacy and security.
- **Fine-grained Access Control**: Through rigorous access control, **EncChain** enforces precise user permissions for knowledge bases. By defining roles like "questioner" and "knowledge base owner" and assigning access based on unique identifiers for these roles, it mitigates unauthorized data access and potential exfiltration.
- **Streamlined Integration and Application**: As a Python package, **EncChain** offers straightforward integration into third-party applications, facilitating adoption by allowing developers to easily incorporate its features. This ease of use, combined with support

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685888

for both encrypted and plaintext queries, significantly reduces the complexity for developers new to the system.

• **Execution Safety in Trusted Environments**: **EncChain** and its associated LLMs are deployable within trusted execution environments, leveraging advanced hardware security features to safeguard virtual machine memory privacy and integrity. This setup ensures that sensitive data is shielded from both the host operating system and the virtual machine manager, enhancing operational security.

• **Remote Attestation for Enhanced Trust**: **EncChain** enables the use of remote attestation technologies to confirm the security and trustworthiness of the execution environments for itself and the deployed LLM, providing users with additional confidence in the security measures of LLM applications.

2 PRELIMINARIES

Retrieval Augmented Generation. RAG [3] architecture represents a significant advancement in addressing the challenge of hallucination in LLMs, emerging as a dominant pattern in developing LLM applications, particularly enhancing logical reasoning and data comprehension from private knowledge bases to augment question-answering (QA) capabilities. It is pivotal in scenarios like knowledge-based questioning and intelligent assistance. The RAG framework involves segmenting private knowledge into embedding vectors stored in a database. Upon receiving a question, the system converts it into a vector, retrieves the most relevant knowledge via vector similarity search, and merges this with the question to form a comprehensive prompt for LLMs.

Trusted Execution Environment. TEEs [4, 5] provide a cornerstone technology by offering secure and isolated execution spaces within processors, enhancing the security of data and code against potential threats from compromised operating systems or hypervisors in the complex landscape of cybersecurity and data privacy. Within this spectrum, Intel's Trust Domain Extensions [2] (TDX) serve as an evolved form of TEEs, tailored to bring their benefits into the realm of virtualization. TDX introduces the concept of trusted domains, in which virtual machines operate in isolation with hardware-level protections. This innovation directly addresses the intricate challenges of maintaining data privacy and security in environments such as cloud computing and data centers.

3 EncChain SOLUTION

3.1 Threat Model

The RAG architecture in QA leads to two primary threats: unauthorized access and data exfiltration. Firstly, its reliance on plaintext storage of knowledge bases and user queries permits developers unfettered access, creating a vector for data leaks in cases of malicious intent or system compromise. Secondly, the architecture lacks rigorous access controls, enabling users to potentially retrieve sensitive information beyond their clearance through intentionally designed queries. These threats collectively jeopardize data integrity and confidentiality, necessitating an immediate implementation of enhanced security protocols to mitigate the risks of unauthorized access and ensure the privacy protection of LLM applications.

3.2 Architecture Overview

The **EncChain** architecture, delineated in Figure 1 for LLM application deployment, emphasizes security and operational integrity.

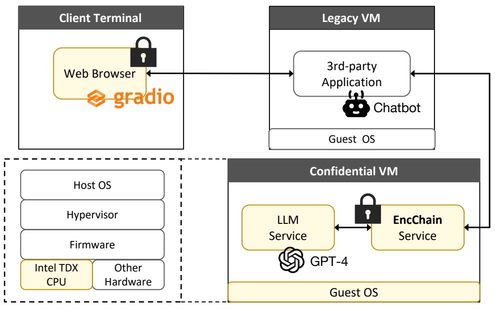


Figure 1: The architecture of the EncChain demonstration.

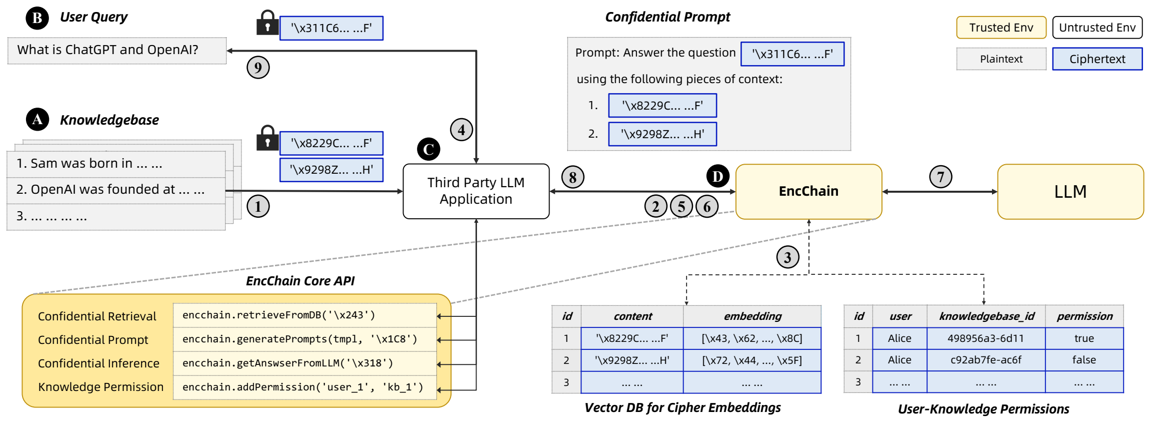
It treats the client terminal as secure, encrypting data before it exits, protecting it during transmission. Third-party applications are hosted on virtual machines (VMs), establishing a clear operational divide. **EncChain** and its models operate within secure virtual environments utilizing advanced VM technologies like TDX for enhanced runtime security. These environments are reinforced by hardware security extensions, safeguarding virtual memory from unauthorized access by the host OS and hypervisor. Third-party applications leverage **EncChain**'s APIs for encrypted data interactions and secure business logic development. Remote attestation technology allows users to verify the security of **EncChain** and LLM environments, adding a layer of trust. **EncChain**'s security protocol includes data encryption at domain entry and exit, strict access control, and the synergistic use of secure VMs and remote attestation, providing a robust framework for secure LLM application deployment, addressing the critical need for data security.

3.3 Fine-grained Knowledge Control

EncChain enhances privacy attributes in LLM applications using RAG-based private knowledge base inference through the key action of leveraging fine-grained knowledge control. This innovation, derived from Operon's privacy-protected data management [6], embodies the concept of the Behavior Control List (BCL). Specifically, **EncChain** allows "knowledge owners" to establish a binary relationship between the "questioners" and the "knowledge bases." Upon the questioner posing a question, triggering the LLM's inference, **EncChain** ensures that the search for relevant knowledge vectors occurs exclusively within an authorized subset of vector databases, generating answers based on this relationship. It solves the issue traditionally addressed either by employing multiple distinguished LLM instances to segregate knowledge for privacy protection (sacrificing efficiency and increasing costs) or by utilizing a single system but facing privacy risks. **EncChain**'s innovation lies in its ability to protect privacy while optimizing the retrieval and integration process of knowledge, thereby finding an effective equilibrium between privacy security and knowledge utilization.

3.4 System Workflow

We present the procedural workflow of **EncChain** through a specific example, as illustrated in Figure 2. In this scenario, we assume four distinct roles: Ⓐ knowledge base data owners; Ⓑ questioners; Ⓒ third-party software developers providing QA applications; and Ⓓ TEEs (e.g., cloud infrastructure) for deploying LLMs with **EncChain**. We note that, in practical scenarios, Ⓐ and Ⓑ might represent the same entity, or Ⓑ could be a controlled party of Ⓐ (for



Ⓑ User Query
What is ChatGPT and OpenAI?
'\x311C6... ...F'
Ⓐ Knowledgebase
1. Sam was born in ... ...
2. OpenAI was founded at ... ...
3. ... ...
'\x8229C... ...F'
'\x9298Z... ...H'
Ⓒ Third Party LLM Application
Confidential Prompt
Prompt: Answer the question '\x311C6... ...F'
using the following pieces of context:
1. '\x8229C... ...F'
2. '\x9298Z... ...H'
Trusted Env
Untrusted Env
Plaintext
Ciphertext
Ⓓ EncChain
LLM
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
EncChain Core API
Confidential Retrieval encchain.retrieveFromDB('\x243')
Confidential Prompt encchain.generatePrompts(tmpl, '\x1C8')
Confidential Inference encchain.getAnswserFromLLM('\x318')
Knowledge Permission encchain.addPermission('user_1', 'kb_1')
id content embedding
1 '\x8229C... ...F' [\x43, \x62, ..., \x8C]
2 '\x9298Z... ...H' [\x72, \x44, ..., \x5F]
3 ... ...
Vector DB for Cipher Embeddings
id user knowledgebase_id permission
1 Alice 498956a3-6d11 true
2 Alice c92ab7fe-ac6f false
3 ... ...
User-Knowledge Permissions

Figure 2: An illustrative workflow of the EncChain. For simplicity, the figure omits ⓪, which indicates the initialization phase.

instance, a sales Employee B of Company A authorized to inquire about product information and receive answers, but not permitted to ask questions related to the company's finances). Similarly, Ⓒ and Ⓓ could also be consolidated into a single entity, offering both software applications and compute resources, depending on the service model. Upon establishing the identities of the involved roles, the workflow proceeds as follows:

⓪: Ⓐ first verifies the **EncChain** instance by utilizing TEE remote attestation, ensuring that **EncChain** operates with confidentiality and integrity. This enables the handing over of keys to **EncChain**, allowing it to decrypt ciphertext within the TEE.

①: Ⓐ encrypts knowledge bases and uploads them to Ⓒ, ensuring that even a malicious Ⓒ cannot comprehend the hosted info.

②: Ⓒ further hands over the uploaded knowledge bases into the trusted domain via **EncChain**'s APIs.

③: The uploaded knowledge bases are decrypted using the owners' key, vectorized, and securely stored in the vector database.

④: Ⓑ poses a question that is encrypted with its own key before being submitted to Ⓒ. Similar to ⓪, Ⓑ also needs to submit its key for **EncChain** to interpret its question upon attestation.

⑤: Ⓒ, unable to understand the question submitted in ④, can only retrieve relevant contextual knowledge through **EncChain**'s Retrieve interface. Notably, at this stage, **EncChain** delineates the appropriate subset of knowledge bases for Ⓑ's query based on *User-Knowledge Permission*. Knowledge vectors pertinent to the question, retrieved by Ⓓ, are returned to Ⓒ encryptedly.

⑥: Ⓒ, following the desired business logic, constructs an appropriate prompt and requests **EncChain** for LLM inference.

⑦: Ⓓ decrypts the request's ciphertext by the questioner's key of Ⓑ within the TEE and carries out the LLM inference process.

⑧: Ⓓ returns the model inference output to Ⓒ in encrypted form.

⑨: Ⓒ returns the encrypted response to Ⓑ, who then uses its own key to decrypt and obtain the answer to the question.

4 DEMONSTRATION

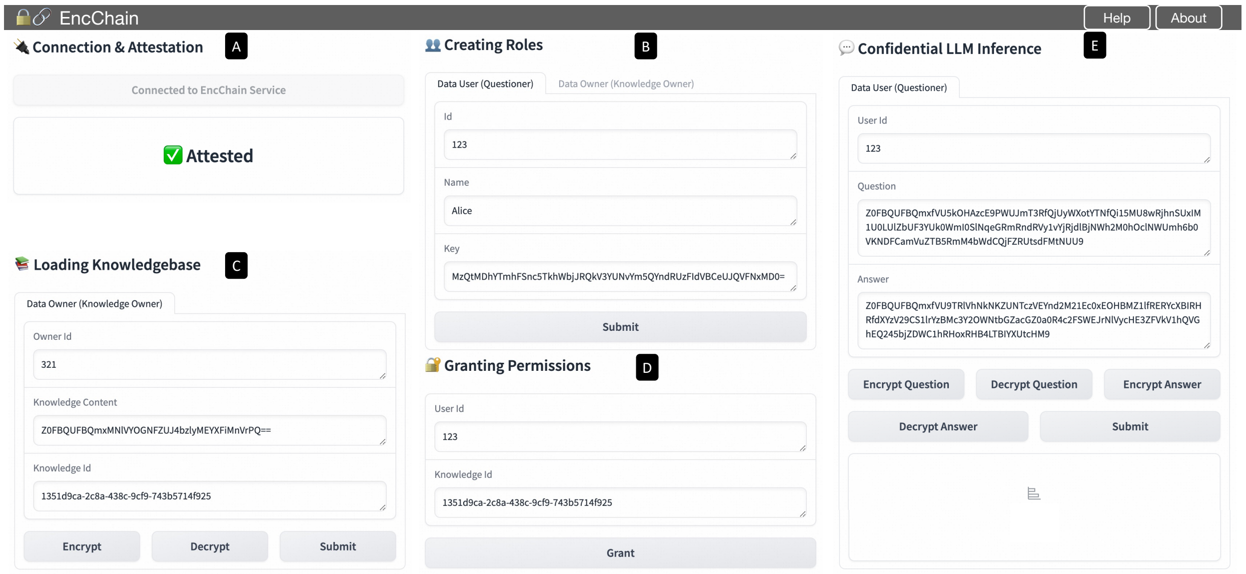
During the demonstration, we will present to the audience a comprehensive, end-to-end framework for live deployments of LLM applications, emphasizing privacy safeguards for proprietary knowledge bases and rigorous permission control, facilitated by **EncChain**.

```
 1 from EncChain import OpenSourceApp
 2 # Create a chatbot instance with built-in opensource LLM
 3 app = OpenSourceApp(permissions=permissions.db, ...)
 4
 5 # Register a knowledge base owner with its owner's key
 6 owner_id = app.permissions.add_owner(owner_name, key)
 7
 8 # Insert a knowledge base into EncChain's vector database
 9 kb_id = app.add_kb(data_type, enc_knowledge_body)
10 # Designate the ownship of the hosted knowledge base
11 app.permissions.add_kb(owner_id, kb_id)
12
13 # Register a questioner with its user's key
14 user_id = app.permissions.add_user(user_name, key)
15 # Grant permissions to the user for the knowledge base
16 app.permissions.add_policy(user_id, kb_id, owner_sig)
17 # Answer the question encryptedly with the user's key
18 app.query(enc_question, user_id)
```

Figure 3: Usage examples of the EncChain Python Library.

4.1 Python Library

First, we elucidate the process by which backend developers can adeptly and seamlessly initiate a hardware-secured LLM instance, utilizing the Python library provided by **EncChain**, complemented by illustrative code excerpts in Figure 3. **EncChain** offers a flexible, modular design, while also allowing for the instantiation of a confidential LLM instance in its default mode (line 3), which constructs a built-in open-source model. Due to space constraints, we omit a detailed discussion of additional construction parameters, such as those based on an existing permission database. Data owners should, under the assurance of instance trustworthiness, submit their keys to the instance's permission table (line 6). Subsequently, data owners can host their privately held knowledge on **EncChain** (line 9), encrypted with the owner's key, for future model inferences during QA sessions. Specifying the ownership of knowledge bases is essential for **EncChain** to determine which key to use for decrypting knowledge within the TEE for embedding and further knowledge integration (line 11). Similarly, questioners, upon verifying the instance's credibility, should submit their unique keys (line 14). These keys are utilized to decrypt their encrypted queries and to encrypt answers to their questions. Before posing questions, it is imperative to ensure that users are authorized to use specific knowledge bases as context for generating answers, a process that requires the owner's signature for authorization (line 16). Thereafter, users may encrypt their queries using the previously



Owner Id: 321

Knowledge Content: Z0FBQUFBQmxMNlVYOGNFZUJ4bzlyMEYXFiMnVrPQ==

Knowledge Id: 1351d9ca-2c8a-438c-9cf9-743b5714f925

Id: 123

Name: Alice

Key: MzQtMDhYTmhFSnc5TkhWbjJRQkV3YUNvYm5QYndRUzFIdVBCeUJQVFNxMD0=

User Id: 123

Knowledge Id: 1351d9ca-2c8a-438c-9cf9-743b5714f925

User Id: 123

Question: Z0FBQUFBQmxfVU5kOHAzcE9PWUJmT3RfQjUyWXotYTNfQi15MU8wRjhnSUxIM1U0LUlZbUF3YUk0WmI0SlNqeGRmRndRVy1vYjRjdlBjNWh2M0hOclNWUmh6b0VKNDFCamVuZTB5RmM4bWdCQjFZRUtsdFMtNUU9

Answer: Z0FBQUFBQmxfVU9TRlVhNkNKZUNTczVEYnd2M21Ec0xEOHBMZ1lfRERYcXBIRHRfdXYzV29CS1lrYzBMc3Y2OWNtbGZacGZ0a0R4c2FSWEJrNlVycHE3ZFVkV1hQVGhEQ245bjZDWC1hRHoxRHB4LTBIYXUtcHM9

Figure 4: Screenshot of a sample chatbot application using EncChain.

registered key, with the entire LLM inference process remaining protected within the TEE (line 18). Answers are then encrypted using the same key and returned to the user.

4.2 Web User Interface

Second, we will display a browser-based user interface for end-users, as depicted in Figure 4, which focuses on the secure integration of private knowledge bases into the encrypted LLM inference processes, ensuring a trusted deployment environment. Similar to Section 4.1, initiating QA sessions via the web interface necessitates completing essential preliminary steps, including **A** connecting to the service instance and performing remote authentication, **B** creating user roles, **C** uploading knowledge bases, and **D** granting permissions for users' access to specific knowledge bases. Following these preparatory activities, users can engage with **E** for privacy-protected third-party QA applications. The interface reveals that both queries and their corresponding responses are encrypted throughout the entire data pipeline, effectively eliminating the possibility of privacy breaches, with decryption to plaintext occurring only on the user's client side. Of course, this explicit encryption and decryption on the client side is for demonstration purposes and could be transparent to users.

4.3 Implementation Details

The demonstration encompasses approximately 2000 lines of Python code, utilizing the LangChain framework[1] for optimized data segmentation and retrieval. It also incorporates Chroma DB[2] for text vectorization and enhanced similarity search capabilities, GPT4All[3] for local open-source LLM inference, and Gradio[4] for the web interface. The deployment infrastructure for this demonstration is hosted on Alibaba Cloud, utilizing both regular ECS instances and TDX-enhanced confidential VM instances. Regarding performance, the end-to-end latency experiences a 0.3%-1.5% increase, which mainly results from the data encryption/decryption and module-to-module data transmission within **EncChain**. This amalgamation of technologies and methodologies signifies a holistic strategy in constructing a secure, efficient, and scalable infrastructure for sophisticated data processing and analytical endeavors.

[1]https://github.com/langchain-ai/langchain
[2]https://github.com/chroma-core/chroma
[3]https://github.com/nomic-ai/gpt4all
[4]https://github.com/gradio-app/gradio

5 CONCLUSION

EncChain emerges as a vital enhancement in addressing privacy concerns within LLM applications, especially tailored for RAG-based private knowledge deployment scenarios. It uniquely combines high-quality model inference, flexible delegated computation, robust privacy protection, and ease of use into a cohesive solution. This innovation not only strengthens the privacy framework for LLM applications but also facilitates their scalable and secure integration into clouds. Thereby, it enables organizations to leverage advanced AI capabilities without compromising data security.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, et al. 2020. Language Models are Few-Shot Learners. In *NeurIPS*, Vol. 33. 1877–1901.

[2] Intel. 2023. White Paper: Intel Trust Domain Extensions. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html.

[3] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, Vol. 33. 9459–9474.

[4] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted Execution Environment: What It is, and What It is Not. In *TrustCom/BigDataSE/ISPA (1)*. IEEE, 57–64.

[5] Mo Sha, Jialin Li, Sheng Wang, Feifei Li, and Kian-Lee Tan. 2023. TEE-based General-purpose Computational Backend for Secure Delegated Data Processing. *Proc. ACM Manag. Data* 1, 4 (2023), 263:1–263:28.

[6] Sheng Wang, Yiran Li, Huorong Li, Feifei Li, Chengjin Tian, Le Su, Yanshan Zhang, Yubing Ma, Lie Yan, Yuanyuan Sun, Xuntao Cheng, Xiaolong Xie, and Yu Zou. 2022. Operon: An Encrypted Database for Ownership-Preserving Data Management. *Proc. VLDB Endow.* 15, 12 (2022), 3332–3345.