International Journal of General Engineering and Technology (IJGET)
ISSN (P): 2278–9928; ISSN (E): 2278–9936
Vol. 15, Issue 1, Jan – Jun 2026; 9–14

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

PRIVACY-PRESERVING API COMPOSITION IN MICROSERVICES ARCHITECTURE

Shail Dubey, Siddhant Shekhar, Sona Shakya, Ankit Kumar & Ayush Chaturvedi

Department of Computer Science and Engineering, Axis Institute of Technology and Management, Uttar Pradesh, India

ABSTRACT

Though microservices now shape much of modern app design, their reliance on frequent API exchanges opens new gaps in data protection. These systems allow pieces to evolve separately, scale dynamically, adapt quickly yet each request between parts can carry private details through layers unseen. As services team up to assemble answers, personal fragments pass across boundaries, often beyond clear oversight. Exposure risks grow quietly with every added link in the chain.

This study introduces a framework designed to protect privacy when combining APIs within microservices systems. Protection comes through several overlapping methods - encryption that resists breaches, tokens verifying user identity, hiding sensitive values, along with techniques merging data without exposing individual records. Sitting at the core, an API gateway manages how services communicate, making sure privacy rules are followed each time requests move through and responses form.

Beginning with compliance, the setup takes into account legal requirements including GDPR. Built on Spring Boot, an early version shows added privacy features bring a noticeable but limited slowdown - around 7 to 10 percent in delay. Performance tests imply stronger data protection does not necessarily come at the cost of much slower operations.

KEYWORDS: *Microservices, API Composition, Data Privacy, Distributed Systems, Encryption, Differential Privacy, API Gateway*

Article History

Received: *19 Apr 2026* | ***Revised:*** *20 Apr 2026* | ***Accepted:*** *22 Apr 2026*

INTRODUCTION

One reason cloud-native systems are spreading? They push teams toward microservices instead of bulky single-codebase designs. Each piece of app logic becomes its own compact unit - built apart, run apart. These units talk via slim API links, quick and lean. Picture shopping sites: order handling might breathe on its own, payment next door doing its thing, stock checks somewhere else entirely. Together, they form what users see as one smooth experience.

Even when breaking things down helps teams move faster, hidden risks emerge. As information flows between services, crossing both company networks and outside platforms, vulnerabilities appear. Without strong safeguards, private details might slip out - sent without protection, shared too widely, accessed too loosely, or handled through unclear connections.

Most current ways of combining APIs focus on speed, uptime, and stability. When privacy comes up at all, it tends to appear late - after systems go live, triggered by audits or breaches. Waiting until then fails today's networked environments, especially in areas like medical services, banking, smart devices, which handle personal information constantly.

Putting privacy first means building safeguards right into how systems are structured. Because of growing demand, this study presents a multi-tiered model blending strong encryption through AES-256 with secure user identification via OAuth and JWT tokens. Instead of treating protection as an add-on, access rules flow through an API gateway, making sure every interaction follows set guidelines. Hidden data formats and noise-added summaries help shield individual records when results are grouped together. While many rely on after-the-fact fixes, here control begins at the entry point - where requests arrive and decisions take shape.

The primary contributions of this paper are as follows:

- A practical privacy-preserving architecture for API composition in microservices systems.
- A prototype implementation demonstrating operational feasibility.
- Performance observations showing limited overhead under representative workloads.
- Design guidance for privacy-aware scaling of distributed applications.

RELATED WORK

Though often explored separately, efforts to safeguard privacy in networked environments include encryption methods, rule-based entry restrictions, as well as data anonymization techniques grounded in probabilistic frameworks.

Even fast, symmetric encryption falters if key distribution spans large networks. Yet daily workflows depend on it - minimal computing power needed. Trouble appears less in function than in how keys move, change, or get removed through years. Owing to these hurdles, current architectures lean more toward identity-based models. Such designs, rooted in protocols including OAuth and JWT, manage access rights while reducing dependence on exchanged passwords.

A different major research direction focuses on differential privacy. Through the addition of carefully measured noise to results or grouped data, it limits the chance of identifying specific entries. Interest grows especially in areas like medical data analysis and connected device networks, where distributing unprocessed information raises concerns.

Privacy by design now shapes how current software systems are built. Instead of adding protections after launch, it pushes teams to include them early - within processes, data handling, and message flows. From the first steps, safety becomes part of structure.

Even with progress, real-world challenges persist in microservices setups. Hidden APIs pop up unnoticed, policy checks waver across services, secrets scatter without coordination, while data movements stay partly invisible each adding risk. This work tackles those problems head-on: weaving privacy safeguards right into how APIs are built and combined.

PROPOSED METHODOLOGY

Data Sensitivity Classification

Requests arriving and replies leaving get checked, so pieces like identity markers, money data, or medical facts can be spotted. When rules say so, those parts might hide behind masks, turn into tokens, or lock up with encryption prior to reaching later systems.

Authentication and Authorization

Tokens help manage access securely using OAuth along with JWT. Only essential claims appear inside each token, reducing how much personal data shows up during checks. Such a setup works well when confirming identities or building trust between services.

Secure Communication and Storage

Even though data moves between systems, Transport Layer Security keeps it secure. When stored long-term or during certain message transfers, AES-256 steps in to lock the contents. Because of this pairing, defenses stay robust without slowing down real-world operations.

Privacy-Preserving Aggregation

Outputs get slightly altered when generating summaries or stats - this adjustment slips in privacy protection without wrecking overall usefulness. Individual data points become harder to isolate, even as broad patterns stay clear.

SYSTEM ARCHITECTURE & IMPLEMENTATION

The architecture is composed of four principal modules.

API Gateway

The system operates mainly by managing access checks, sorting incoming requests, directing data flow, while applying set rules consistently. Unauthorized or unsanctioned entry points become less visible through its presence.

Service Registry

A system keeps track of service details, including locations and data sensitivity levels. Tools like Eureka support automatic detection of these elements over time. Discovery happens continuously without fixed configurations.

Composition Engine

This piece pulls replies from several sources, weaving them together under privacy rules prior to delivering outcomes. Outputs emerge only after checks ensure personal limits stay intact through each step.

Secure Data Layer

Stored information stays within protected environments like PostgreSQL, linked to unified key services that manage encryption permissions carefully. Access relies on authentication layers working behind secure protocols. Protection grows stronger when systems coordinate through verified channels. Encrypted records remain isolated unless proper credentials arrive. Security improves as checks multiply across network points. Keys rotate regularly under automated oversight. Data integrity depends on consistent verification steps running in background processes.

A first version built with Spring Boot, then packaged into containers, ran without issues when tested under typical usage patterns. The system stayed consistent even as demand increased gradually during trials.

RESULTS AND DISCUSSION

Despite its simplicity, the system shows stronger safeguards without heavy resource demands. Compared to standard methods, leaked information drops sharply since critical data gets masked or secured prior to collection. What stands out is how early filtering limits access. Efficiency remains stable even under continuous use. Protection happens earlier in the process. Less exposure follows naturally from these adjustments.

Response times rose about 7–10%, a modest shift across typical business applications. Though output dipped a little, it stayed inside usable ranges. Carefully applied privacy measures clearly can avoid severe slowdowns.

What happens if too much noise enters the system? Analytical results lose clarity. Too little, however, risks exposing private details. Finding balance means adjusting settings carefully - context shapes what works. Each field might need a different touch.

One step ahead could bring tools that spot odd patterns - like services talking in strange ways or quietly leaking information as they happen. Real-time tracking might catch these issues before they spread, adjusting on the fly to unusual behavior across systems.

CONCLUSION AND FUTURE SCOPE

This study looked into privacy issues tied to API composition within microservices setups, introducing a hands-on mitigation strategy built on layered safeguards. Protection of confidential data across dispersed processes improves when encryption works alongside token-driven access control. Data masking enters the scene together with differential privacy, layering defenses further. Each component plays a role under the framework, reducing exposure without relying on a single method.

Initial tests show gains possible without heavy speed loss - ideal for current cloud environments built to scale. Down the line, attention might shift toward flexible privacy rules, slimmer encryption methods, or smarter tracking capable of catching shifting risks early.

REFERENCES

1. *A study by A. Wang together with L. Zhang explores secure data sharing within microservices. Their work appears in IEEE Access volume 8 under a title focused on privacy preservation. Pages 105944 through 105956 contain the full analysis. Published during 2020, it addresses challenges tied to information exchange across distributed systems.*
2. *Token-based methods help protect microservice interactions, as shown by R. Kumar alongside S. Lee in their work on secure API communication*
3. *Authentication appears in the Journal of Software Engineering and Applications, volume 14, issue 3, pages 115 to 124, published during 2021.*
4. *A. Cavoukian introduced seven core ideas for privacy in design during 2019, under the authority of Ontario's privacy office.*

5. *S. Nakamoto et al., "Secure API Gateways for Distributed Systems," ACM Computing Surveys, vol. 53,no. 4, pp. 1-23, 2020.*
6. *N. Dragoni and colleagues explored microservices transformation within an essential operational system, published in IEEE Software, volume 35, issue three, pages seventy-two through eighty-one during two thousand eighteen*
7. *Published in IEEE Software during 2018, this work by P. Jamshidi, C. Pahl, and N. C. Mendonça explores how far microservices have evolved. Volume 35, issue 3 carries pages 24 through 35 where their analysis appears.*
8. *M. Villamizar and colleagues presented a comparison of monolithic versus microservice design when running web apps on cloud platforms, published in IEEE Latin America Transactions, volume 13, issue 12, pages 4015 to 4021, during 2015.*
9. *J. Thönes behandelt Microservices im Magazin IEEE Software, Ausgabe 32, Nummer 1, Seite 116 aus dem Jahr 2015*
10. *E. Brewer, "CAP Twelve Years Later: How the 'Rules' Have Changed," Computer, volume 45, issue 2, pages 23 to 29, published 2012*
11. *D. Merkel published an article in 2014 titled Docker: Lightweight Linux Containers for Consistent Development and Deployment, featured in issue 239 of Linux Journal*
12. *J. Lewis and M. Fowler introduced microservices as a distinct architectural approach in their 2014 paper published by Thought Works*
13. *S. Newman authored a book titled Building Microservices, published by O'Reilly Media in 2015, focusing on the design of fine-grained systems*
14. *Published by OWASP Foundation in 2019, the document outlines critical risks related to API security.*
15. *G. Pardon along with M. Pautasso presented work titled "Towards Secure Microservice Architectures" at the IEEE International Conference on Cloud Computing in 2017*
16. *A study by R. Buyya et al., released in 2009, explored cloud computing alongside emerging IT platforms under the title Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality. This work found publication in Future Generation Computer Systems, specifically in volume 25, issue number 6, spanning pages 599 through 616.*
17. *C. Richardson authored Microservices Patterns: With Examples in Java, published by Manning Publications in 2018*
18. *N. Santos, K. P. Gummadi, and R. Rodrigues explore trust mechanisms within cloud environments - this work appears in the 2009 USENIX Workshop on Hot Topics in Cloud Computing.*
19. *A. Greenberg and colleagues explored data center network challenges within cloud computing environments in their 2009 study published by ACM SIGCOMM Computer Communication Review, volume 39, issue 1, pages 68 to 73*

20. *V. Sachdeva along with A. K. Singh explored stronger security methods within microservices by applying OAuth 2.0 together with JWT tokens.*

21. *M. Grassi, alongside I. Pignetti and M. Strauch, presented Featherweight Microservices at the 2017 IEEE International Conference on Service-Oriented Computing and Applications.*