

A Multi-Functional Content Distribution Framework Centered on Edge Authentication

Siyuan Chen
Shanghai Nanyang High School
Shanghai China
chensiyuan20080908@126.com

Abstract

The paper proposes a framework solving problems of traditional centralized data distribution architectures. Those architectures suffer from high server load, high latency and tightly bound data and authentication logic. The goal is not only faster computation, but also smarter distribution and management. It takes advantage of edge computing and separates the outermost layer of security control from traditional servers. That can prevent origin servers from being exposed to the public network. Small and medium scenarios such as school clubs and public facilities can benefit from this solution. It works like a small server, and can reduce server load by processing simple requests like invalid requests, high-frequency AI data collection and suspicious requests. The framework features distributed token check, dynamic routing and multi-scenario data distribution. The framework needs a platform key, database and customized routing rules. It can also run without a database. The framework can be enhanced by adding custom extensions. This framework is applicable to dynamic password distribution (e.g., offline software authentication), limited data distribution (e.g., school club file sharing), IoT device authentication (e.g., smart sensors in classrooms), or serving as an API gateway (e.g., transforming map APIs to a unified format).

CCS Concepts

• Security and privacy → Security services → Authentication
• Computer systems organization → Architectures → Distributed architectures • Information systems → World Wide Web → Web services

Keywords

Edge Computing; Edge Authentication; Token-Based Access Control; Content Distribution; Distributed Architecture

ACM Reference format:

Siyuan Chen. 2026. A Multi-Functional Content Distribution Framework Centered on Edge Authentication. In *Companion of the 2026 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp Companion '26)*, October 11-15, 2026, Shanghai, China. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3798063.3837832>



This work is licensed under a Creative Commons Attribution International 4.0 License.

UbiComp Companion '26, October 11-15, 2026, Shanghai, China.

© 2026 Copyright is held by the owner/author(s).

ACM ISBN 979-8-4007-2533-3/2026/10.

<https://doi.org/10.1145/3798063.3837832>

1 Introduction

Traditional server frameworks suffer from these common problems: all requests are processed by central server(s), important data is tightly bound with authentication logic, and huge amounts of meaningless requests can use up server resources. With the development of edge computing, a lightweight, flexible and edge-friendly alternative is available now.

This paper presents a Lightweight Multifunctional Content Distribution Framework Centered on Edge Authentication. The main idea is to reduce the workload of the server to verify tokens and move the duty to edge computing. A client typically carries a token. The edge worker verifies the token and returns data or configuration.

The framework is:

1. **Lightweight.** Typically, no database, no session management.
2. **Extensible.** System functions can be easily added or modified without changing the verification core.
3. **Non-intrusive.** The original server only needs to store content and simple authorization logic for edge workers.

Typical use cases include software license verification, dynamic license or key publish, limited files sharing, configuration sending for IoT devices, and paid private package distribution. With online shop system, keys can be easily generated and sent. In this framework, “content” refers not only to static files, but also to authenticated configuration data, lightweight web applications, API responses, service endpoints, and dynamically generated resources

2 Related Work

2.1 Edge Authentication Systems

1. **Cloudflare Zero Trust Access** and similar providers use stateless JWT in verification process but rely on external identity providers and policy management. In contrast, the proposed framework performs edge verification with independently generated tokens, requiring no external identity infrastructure.
2. **Tencent EdgeOne** and similar providers forward requests to a customer-specified server for verification [1], and authorization logic remains external. The proposed framework handles both edge verification and authenticated content distribution without centralized authorization infrastructure.

2.2 API Gateway and Request Adaption Architectures

Although acting as an API gateway is only a small part of this framework, its deployment goals differ from popular gateways like Kong. The proposed framework focuses on lightweight edge-native deployment, edge-side verification, and authenticated content distribution.

2.3 Edge-based Page Hosting services

The framework can host pages as well. In contrast to edge-based Page Hosting services such as Cloudflare Pages and Tencent EdgeOne Pages, the proposed framework generates pages dynamically based on the user's token and routing rules, optionally combining local and remote data.

3 System design

3.1 Implementation

The framework is implemented in JavaScript and deployed on Cloudflare Workers, which provides a lightweight serverless execution environment. [2] Each widely deployed worker contains the same token verification function, and it can judge what to do based on user group, requested method and path. Tokens typically include user identity, expiration time, and a user group code, encrypted using a replaceable encryption method. The worker acts as a bridge between users and the origin server while filtering suspicious requests before they reach the origin.

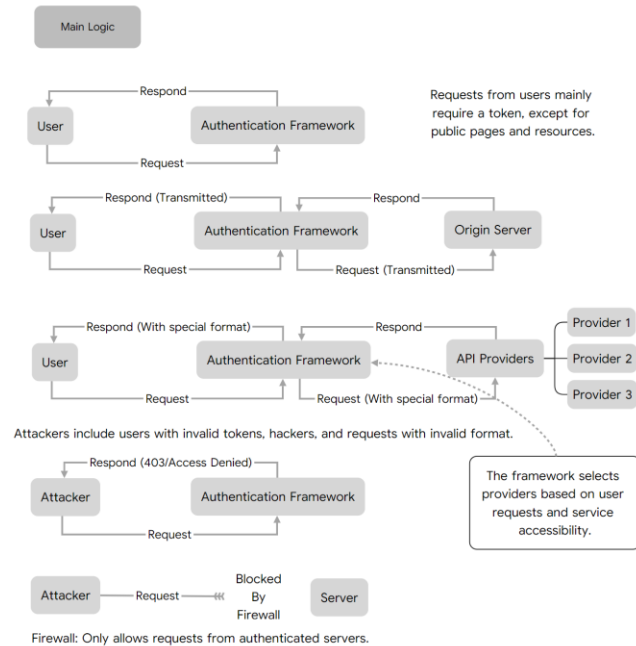


Figure 1: The main logic of the framework.

3.2 Architecture

The framework consists of three layers:

1. **Client layer:** users interact with the edge layer to obtain data. Each request must carry a token.
2. **Edge layer:** receives all requests. It directly responds to requests that do not need to reach the origin (like returning 403 to invalid tokens), and forwards valid requests to the origin server.
3. **Origin layer:** stores all data. It only accepts requests from the authenticated edge layer.

This design ensures the security of the origin server. Direct access to the origin is prevented because all traffic passes through the edge worker.

3.3 Extensibility

The framework is designed to be easily extended for each group of users or globally.

1. **Replaceable encryption.** In the current version, RC4 is used for encrypting and decrypting tokens with the platform key. This choice ensures compatibility across different serverless environments without additional cryptographic dependencies. RC4 is currently used primarily for lightweight compatibility considerations and should be replaced by stronger algorithms such as AES-GCM in real-world deployments. (RC4 is used only for prototype compatibility considerations and is not recommended for production deployment.)
2. **Functional extension.** With databases like SQLite, the framework can support user management, dynamic user group switch, token redefinition, and traditional username and password login methods. With counters based on Cloudflare Workers KV or other methods, this framework can limit request rate like 100 times a minute, and also usage quotas like one token can be used only 10 times. Other extensions like API format translator can bring more functions to this framework.

The framework remains lightweight while being ready for more complex scenarios.

3.4 Security Considerations

The proposed framework focuses on moving authentication to the edge layer rather than introducing a new cryptographic protocol. Therefore, its security depends on both the framework design and the deployment environment. In practical deployments, the following considerations are recommended:

1. **Use HTTPS and well-established cryptographic algorithms.** The framework supports replaceable encryption methods, allowing appropriate algorithms to be selected for different deployment scenarios.
2. **Deploy on trusted edge platforms** with mature security practices to provide a secure runtime environment.
3. **Keep software up to date.** When self-hosting services, administrators should promptly apply security updates and avoid using software versions with high security risks.

The framework provides application-level protection through edge-side authentication and request filtering, while runtime security depends on the deployment platform. Regardless of the platform,

protecting secret keys, applying security patches, and preventing malware or unauthorized access remain essential responsibilities of system administrators.

4 Use Cases

4.1 Typical use cases

The framework is normally run with origin server, and optionally with a database for storing information of tokens and other things.

1. **School Club file sharing.** Time-limited tokens are sent to each student, and by using a file storage server or online object storage service like R2 and others, students and guide teachers can share files through this framework with authentication.
2. **Group controlling of sensors.** Each sensor is configured with a unique token, and these tokens include group number (work same as user group). Collected data can be classified and stored in server by its group number, and configurations can be distributed by group.
3. **IoT devices without stable IP address.** Devices use NB-IoT, Redcap, LTE and so on which have no stable IP address can also use the platform for data aggregation and analysis through domain.

Those are the typical use cases of the framework with its basic functions in real-world scenarios.

4.2 Lightweight Edge-only Deployment

Some requests do not require an origin server, and some scenarios can even fully run with the framework only.

1. **Dynamic verification code** (daily or weekly). For offline software, the edge worker generates time-based codes. This removes the need for a separate authentication server.
2. **Token information display and response.** The edge worker decodes a token and returns its data (like user id, expiration, user group) directly to the client. This lets users verify their token without hitting the origin.

By removing the need for an origin server in these cases, the framework lowers deployment complexity and cost. In some cases, this can even save the cost of renting a server.

4.3 API Gateway and Service Adaptation

Centered on edge authentication, this framework is suitable for serving as an API Gateway due to its edge-native acceleration and high extensibility.

1. **Act as a pre-filter for API requests.** Checking tokens or unifying the format of responses and requests. (Serving as a firewall to reduce API costs by rejecting invalid requests before they hit paid backends.)
2. **Integrate multiple backend APIs from different vendors.** API services with different forms can be transformed to unified format. (For example, a new map API becomes accessible, this method integrates it without changing the client.)
3. **Serving as LLM API translation layer.** Providing access to different LLM APIs with varying formats in one platform. (For

example, an old application built for OpenAI's API can be adapted to support other new formats without modifying its code.)

With the ability of format unification, this can also help managers change API sources without updating clients or stopping services for old versions.

4.4 Built in Edge Web Applications

The framework can include a built-in web UI with lightweight edge-side web applications, which can be used to:

1. **Check and manage token status.** With Web Applications, this can be easily done with browser, which is friendly to users.
2. **Host small web applications or functional sites.** This can cut the budget, and simplify deployment.

Some edge-side application functions may need database support, but this will make the platform more functional.

(Figure 2, Figure 3 are captures from the demo website.)

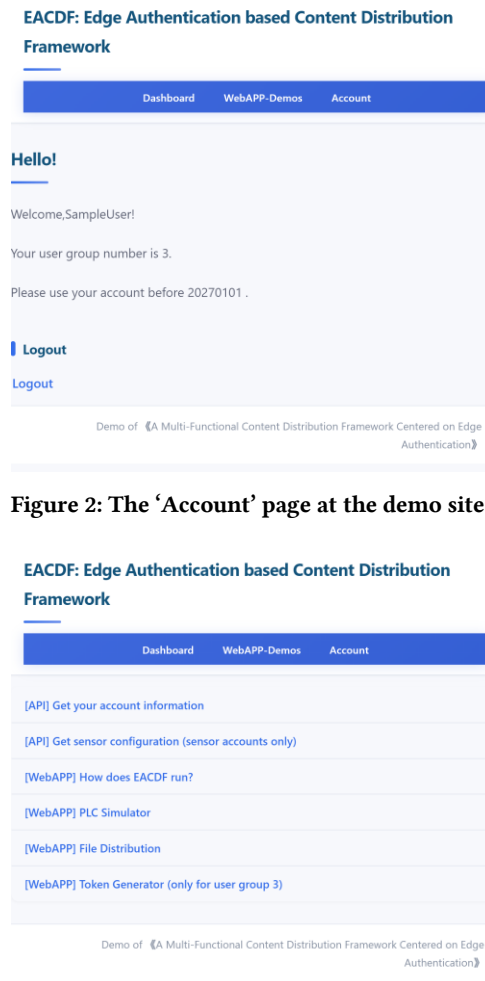


Figure 2: The 'Account' page at the demo site.

Figure 3: The 'WebAPP-Demos' page at the demo site.

5 Implementation and Testing

5.1 Deployment

This framework has been deployed in two modes. In testing, a full deployment includes an edge worker and a local simulated origin server. This setup was used to verify functional correctness and measure request load.

For showcase, the demo runs without an origin server. The edge workers use prepared requests to show the result of the system without relying on external network conditions or backend availability.

5.2 Compatibility

This framework can run properly without modifying code by using any standards-compliant edge computing platform that supports JavaScript and the Web Service Worker API.

Especially in mainland China, using Edge Functions from Tencent EdgeOne, Function Compute from Alibaba, or other similar platforms can increase accessibility and reduce latency.

5.3 Functional Verification and Stability

The framework was tested with 100 API calls (50 with valid tokens and 50 with invalid tokens). In the test environment, the origin server behind the framework received only the 50 valid requests. All invalid requests were blocked by the framework.

In contrast, a traditional server without the edge layer processed all 100 requests.

5.4 Real-world Stability

During real-world deployment, systems built with this framework received thousands of automated probing and unauthorized requests. These requests were filtered at the edge layer before reaching the origin server due to the framework’s edge-side verification architecture. No security or authorization bugs have been reported or discovered during deployment.

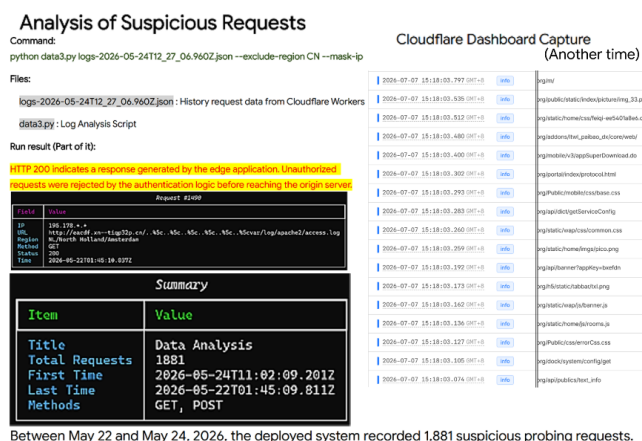


Figure 4: Analysis of Suspicious Requests.

Overall, the framework successfully offloads invalid requests at the edge, reducing the origin server's workload. Four months of real-world operation involving thousands of requests further demonstrated the framework's reliability and deployment stability

6 Conclusion

This paper has presented an edge-based authentication and distribution framework that is easy to extend. It is lightweight and low-cost. Unlike conventional edge proxies, it treats the edge as an active application layer for authentication and content distribution. The current version is based on Cloudflare Workers. The framework was tested with 100 API calls (50 valid tokens, 50 invalid), and all invalid requests were blocked at the edge before reaching the origin. A real system using this framework has been running for four months without issues, serving real users. During that time, it received around 3000 suspicious probes per week, but none had any effect due to the edge-side verification architecture. No security or authorization bugs were found during deployment. The framework also supports replaceable encryption, optional database integration, and rate limiting. It can be used in many scenarios, including campus resource sharing (student clubs), software license authorization, IoT device group authentication, API gateways (like unifying LLM API formats), and industrial device pre-authentication.

Acknowledgements

I thank Dongliang Lu, my teacher at Shanghai Nanyang High School, for trusting me and giving me the chance to explore this project. My thanks also go to my classmates who used the system and reported issues.

I appreciate my parents for their encouragement. Cloudflare Workers and the open-source tools used in this work are gratefully acknowledged.

References

- [1] Tencent Cloud. “Remote Authentication.” EdgeOne Documentation. <https://edgeone.ai/document/57425>. Accessed May 23, 2026.
- [2] Cloudflare, Inc. “Cloudflare Workers.” Cloudflare Docs. <https://developers.cloudflare.com/workers/>. Accessed May 19, 2026.