DYNASEAL: A BACKEND-CONTROLLED LLM API KEY DISTRIBUTION SCHEME WITH CONSTRAINED INVOCATION PARAMETERS

Jiahao Zhao[1], **Jiayi Nan**[3], **Lai Wei**[4], **Yichen Yang**[5]
Xi'an University of Posts and Telecommunications
{zjh, nanjy0108, 1606270965, 2647797263}@stu.xupt.edu.cn

Fan Wu[2]
Xi'an Jiaotong University
wfgods@gmail.com

ABSTRACT

Due to the exceptional performance of Large Language Models (LLMs) in diverse downstream tasks, there has been an exponential growth in edge-device requests to cloud-based models. However, the current authentication mechanism using static Bearer Tokens in request headers fails to provide the flexibility and backend control required for edge-device deployments. To address these limitations, we propose Dynaseal, a novel methodology that enables fine-grained backend constraints on model invocations.

1 Introduction

Large Language Models (LLMs)[1, 2], such as ChatGPT[3], GPT-4 [4], and Claude 3 [5], have shown remarkable progress and impact across diverse domains[6]. Current LLM API access relies on Bearer Token authentication, but this faces challenges with growing edge device inference needs. Edge devices include smartphones, PCs, and microcontrollers interfacing with cloud models.

Two common approaches for edge device model invocation are:

- **Pre-embedded API Keys**: API keys are configured in devices, enabling direct model access.
- **Server Relay**: Intermediary servers relay requests, requiring persistent device-server connections.

Both approaches have limitations: pre-embedded API keys are vulnerable to security breaches, and server relay introduces latency and bandwidth overhead.

Some attempts have been made by the community to address this issue, but each has its limitations. The OpenAI API[7] does not provide server-side keys and can only use Bearer Tokens on the client side. Zhipu AI's keys[8] offer both server-side and client-side invocation methods, supporting server-issued keys and expiration control, but they cannot restrict critical parameters, leaving them vulnerable to attacks. Although OneAPI[9] can redistribute keys, the invocation method remains Bearer Token-based, failing to resolve the client-side invocation problem.

We propose Dynaseal, a novel methodology that enforces backend model invocation constraints. Users retain token-based authentication simplicity while the backend enforces strict controls on model selection and token limits, enhancing model security.

2 Related Work

2.1 Bearer Token Authentication

Bearer token authentication is a widely adopted protocol for securing web APIs and services [10]. This mechanism allows clients to access protected resources by presenting a token, which serves as proof of authorization. The token is typically transmitted in the HTTP Authorization header with the "Bearer" scheme.

2.2 JWT Token

JSON Web Token (JWT) represents a compact, URL-safe means of representing claims between parties [11]. A JWT consists of three parts: a header specifying the signing algorithm[12], a payload containing claims, and a signature for verification. The self-contained nature of JWTs eliminates the need for database lookups, making them particularly efficient for stateless authentication. However, this approach also presents challenges in token revocation and session management, requiring additional mechanisms such as blacklisting or short expiration times.

3 Method

The system architecture comprises three primary components: Large Language Model (LLM) service providers, backend servers, and edge devices.

- The LLM service providers are organizations that host and operate large language models, being responsible for all model-generated responses and inference operations.
- The backend servers function as authentication endpoints for edge devices while also handling business logic implementations. The specific implementation details are determined by the engineering requirements of each deployment scenario.
- Edge devices encompass a diverse range of hardware platforms, from sophisticated devices such as smartphones and personal computers to resource-constrained systems like microcontrollers.

3.1 Backend Authentication

Prior to backend server deployment, a kv-pair (comprising user-id and secret-key) must be obtained from the LLM provider. The kv-pair are subsequently integrated into the service configuration for token generation and identity verification purposes.

3.2 Token Structure

The core mechanism of Dynaseal centers on a specially designed JWT (JSON Web Token) [11], with its structural composition illustrated in Figure 1.

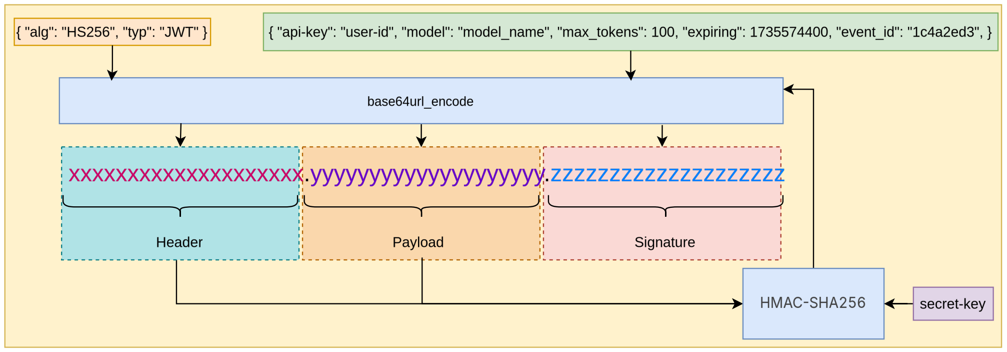
{ "alg": "HS256", "typ": "JWT" }
{ "api-key": "user-id", "model": "model_name", "max_tokens": 100, "expiring": 1735574400, "event_id": "1c4a2ed3", }
base64url_encode
xxxxxxxxxxxxxxxxxxxx.yyyyyyyyyyyyyyyyyyyy.zzzzzzzzzzzzzzzzzzzz
Header
Payload
Signature
HMAC-SHA256
secret-key

Figure 1: Dynaseal token Structure

- **Header**: Declare the encryption algorithm and token type.
- **Payload**: Include key parameters such as model name and maximum token count. The api-key field in payload is configured as user-id of the key-value pair to identify the backend server's identity with the large model provider. The expiration time is set extremely short (e.g., 1s) to prevent reuse.
- **Signature**: Sign with a key-value pair secret-key to ensure token integrity, preventing token tampering.

3.3 Interaction Process

As shown in the 2, the backend server issues tokens to edge devices, with each token encapsulating critical model invocation parameters. Edge devices leverage these tokens to initiate model calls, while the LLM service infrastructure enforces strict invocation constraints based on the parameters embedded within the tokens. Upon completion of the model response, the backend system receives relevant notifications through established callback mechanisms, facilitating comprehensive request lifecycle management.

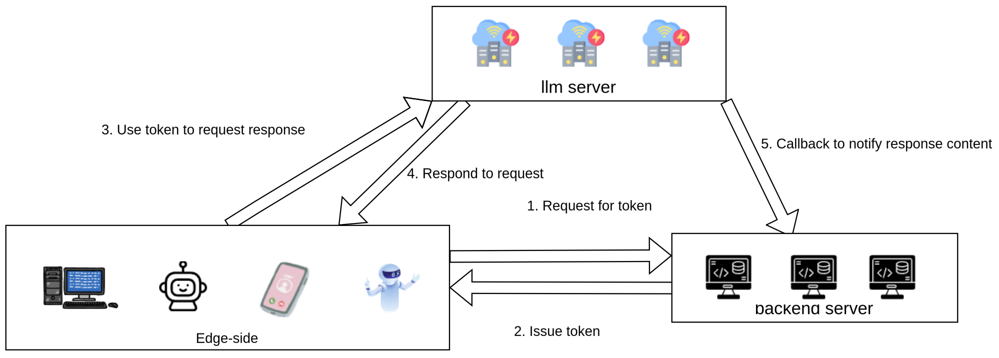

Figure 2: Dynaseal token

1. **Request for token**: Edge-side devices requests token from backend for subsequent interactions accorfind to business logic.
2. **Issue token**: Backend issues token to edge-side.
3. **Request response**: Edge-side uses token to request response insead of Bearer Token.
4. **Respond to request**: Large model provider responds to edge-side.
5. **Callback to notify response content**: Upon response completion, callback notifies response content.

3.4 Attack Prevention

Our system implements comprehensive security measures to prevent potential attacks:

- **Token Tampering**: Malicious actors may attempt to modify token contents to gain unauthorized access. We prevent this through robust **digital signatures** that ensure token integrity, making any unauthorized modifications detectable.
- **Token Replay**: Attackers might try to reuse previously issued tokens. Our system mitigates this risk by implementing **extremely short validity periods**, rendering captured tokens unusable after expiration.
- **Invalid Model Invocation**: To prevent unauthorized model access or parameter manipulation, tokens contain **critical execution parameters**. The LLM service provider enforces strict invocation constraints based on these embedded parameters, ensuring all calls comply with specified limitations.

4 Experiments

4.1 Security Analysis

As shown in Table 1, we compared the advantages and disadvantages of Dynaseal and three other model invocation methods. We evaluated aspects such as client-side key control, tamper resistance, critical parameter control, and multi-model support, all of which are supported by Dynaseal.

Table 1: Comparison of Different Model Invocation Methods

API Provider	**Client-side key control**	**Anti-tampering**	**Critical parameter control**	**Multi-model support**
Openai API	No	No	No	No
Zhipu API	Yes	Yes	No	No
OneAPI	No	No	No	Yes
Dynaseal(Ours)	Yes	Yes	Yes	Yes

4.2 Traffic Consumption Comparison

We compared the network traffic consumption between LLM service providers and backend servers across different approaches including pre-embedded API keys and server relay, as shown in Table 2. The results demonstrate that our method effectively reduces backend server traffic while maintaining constant LLM service provider traffic, simultaneously ensuring key security.

Table 2: Traffic Consumption and Key Deployment Comparison

Method	LLM Provider Traffic	Backend Server Traffic	Client-side Key Pre-deployment
Pre-embedded API Key	Normal	None	Required
Server Relay	Normal	2x	Not Required
Dynaseal	Normal	Minimal	Not Required

5 Conclusion

We propose a novel method, Dynaseal, allowing backend constraints on model invocation, effectively addressing existing edge-side model invocation security issues while avoiding server relay waste. We provide a complete design and interaction flow, demonstrating the feasibility of this approach.

References

[1] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[3] OpenAI. Chatgpt, 2023. `https://openai.com/blog/chatgpt`.

[4] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael,

Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, Cj Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, March 2023.

[5] Anthropic. Claude 3 haiku: our fastest model yet, 2024. Available at: `https://www.anthropic.com/news/claude-3-haiku`.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] OpenAI, December 2024.

[8] Zhipu. Zhipu ai open platform, December 2024.

[9] songquanpeng. Github - songquanpeng/one-api, December 2024.

[10] Michael B. Jones and Dick Hardt. The OAuth 2.0 Authorization Framework: Bearer Token Usage. RFC 6750, October 2012.

[11] Michael B. Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). RFC 7519, May 2015.

[12] Michael B. Jones, John Bradley, and Nat Sakimura. JSON Web Signature (JWS). RFC 7515, May 2015.