Introducing Agent Tokens

A Proposal for Harmonious Coexistence of
Autonomous Agents and Humans on the Internet

Bryson Tang
brysontang@gmail.com

Note: This abstract was generated with the assistance of Anthropic's Claude Opus (June 2024) for clarity in summarization. The content of the paper itself is the author's original work. To see the author's abstract, see Appendix A.

Abstract. As autonomous agents become increasingly sophisticated, the current approach of relying on CAPTCHAs to differentiate between human and AI-generated content on the Internet may prove insufficient. This paper proposes a solution called "agent tokens" to provide a dedicated interface for autonomous agents to interact with websites. By extending HTTP, agent tokens allow autonomous agents to identify themselves and provide structured information about their capabilities, intentions, and provenance. This approach encourages cooperation between websites and agent creators, enabling the labeling and handling of AI-generated content separately from human-generated content. The paper outlines the technical foundations of agent tokens, drawing inspiration from current bot account infrastructure, JSON Web Tokens, and recent research on agent visibility. It also discusses the potential content of agent tokens, including basic information, structure, licensing, training and evaluation details, intentions, and reach. The limitations of the current proposal, such as the need for independent verification, watermarking, and consensus on design, are acknowledged. The paper concludes by emphasizing the importance of providing a channel for autonomous agents to co-exist with humans on the Internet while recognizing the need for further research and infrastructure development to ensure the safe integration of this paradigm shift.

1. Introduction

Autonomous agents—systems that can plan and act independently of human direction (Chan et al., 2024)—will one day roam the Internet freely, and our job is to give artificial intelligence (AI) an interface to interact with us. Our current approach is to lock websites behind a CAPTCHA, betting that there will always be simple tasks that humans can perform that computers cannot. As agents become more sophisticated, the once manageable task might prove impossible. Even without being able to break them, Gao et al. (2022) estimated that the underground user registration market was between $4.8M and $128.1M, meaning you can pay to get around CAPTCHA. Without a proper solution, autonomous agents could flood the Internet with unlabeled AI-generated content, obfuscating humanity from the Internet. Amongst the many dangers this brings to society, the mirage of public opinion is a deep concern and can have a real-world impact. For instance, swarms of autonomous agents manipulate the outcome of elections, undermine democracy, or orchestrate negative sentiment about companies, damaging their image and stock price (Ferrara et al., 2016).

This paper describes a blueprint for an interface that provides autonomous agents with a dedicated channel to interact with the Internet. By doing this, websites can label and handle AI-generated content separately from human-generated content. Agent developers will be encouraged to adopt this standard as it will be less resource-intensive than trying to bypass CAPTCHA. The solution aspires to mirror current Internet standards to fit within the continuity of the Internet's evolution. This paper will also describe what important information agents could provide to humans. Finally, it will acknowledge the shortcomings and infrastructure needed for this solution to be successful. The proposed solution is not just a one-time fix but a flexible and adaptable interface that extends HTTP. HTTP is a good target for this system because it is already how machines communicate with each other and is easily extendable. The hope is that if agents can identify themselves in a standard way over HTTP, websites will benefit by giving them a separate interface. This proposal is not a rigid set of rules but an effort to create a comprehensive solution. The final implementation of this standard should be a collective decision, with implementers picking information to minimize overlap and create a lean set of properties. This process should aim to develop a set of properties where each is a distinct part of the whole.

2. Foundations

Agent tokens are the proposed solution to the issue of agent identification over the Internet. The structure of agent tokens is JavaScript Object Notation (JSON), a list of properties and values; section 2.2 has an example of JSON's structure. Appendix B provides an example of an encoded agent token. This section will outline the three sources of inspiration for agent tokens: Current bot account infrastructure as a social solution, JSON Web Tokens as a technical solution, and *Visibility into AI agents* (Chan et al., 2024) for the content of the tokens.

2.1 Bot Accounts

By providing a less resource-heavy channel to access a website autonomously, those using agents will be encouraged to conform. Reddit's approach to autonomous agents is to set up bot-specific APIs, allowing them to have a dedicated API (Long et al., 2017). They also inform the user when an account is a bot. The hope is that signing up for the API is more accessible than making a bot that can go around the CAPTCHA. X has also begun taking this approach and tags automated accounts with a label (X, n.d.).

Informing the user when interacting with an agent is critical, but providing rich information about the intentions and origins will build more trust. Current systems need to provide this information. Reddit and X's first move also set good momentum toward being open about autonomous accounts. However, collaboratively building universal open standards, instead of everyone having their standard, will allow autonomous agents to flourish across the digital space.

2.2 JSON Web Tokens

This paper combines several concepts from the extension of HTTP to create the pipeline for agent tokens. OAuth 2.0 is the concept that a cryptographic token can verify which user is making the request to the server (Hardt, 2012). These tokens allow the user to perform actions only the particular user can perform, such as making a tweet from their account or changing their password.

Websites can encode user details in these tokens by sending a JSON Web Token (JWT) as the authentication (Jones & Campbell, 2015). Employing JWTs allows the server to make role-based decisions with just the authentication token. When the user provides correct login credentials, the server encrypts and signs the JWT to verify the user. The payload of JWT is JSON, a simple way to structure data in a key pair system, as seen below (Jones et al., 2015).

```
{
	"name": "John Doe",
	"admin": true
}
```

By mimicking known conventions of HTTP, this proposal should fit well into the ecosystem of tools for servers. The goal is to create a system that is not too foreign to how the Internet behaves today while also expressive enough to handle the potential rapid growth of autonomous agents.

2.2.1 Header limits

Headers are additional information about the client sent to the server on each request (Fielding & Reschke, 2014), making them a convenient place to store the agent token. The issue with headers is that they are limited in size. The header limit is usually 8 KB of information, 8192 characters, meaning the agent token would have a size limit while being conscious of other headers' sizes. A solution to this would be limiting the length of the variable-length properties. However, restricting the length constrains the amount of potential information expressed in agent tokens.

The agent token could also be sent to the server once, and then, in exchange for it, a basic authentication token could be received. This flow would set the length limits much higher, as server requests have larger limits than the header. The cost of this approach is that agent tokens would be a less generalized tool. Instead of being able to send the token with each request, first, the server must do a handshake with the agent and then look up the agent token on subsequent calls. This method works but requires additional infrastructure and storage costs on the part of the server. The definition of this exact process is out of the scope of this paper.

2.3 Agent Identification

Chan et al. (2024) detailed three different aspects an autonomous agent could provide about itself to a human. These aspects were used as a framework when designing the content of agent tokens.

- The underlying system
- The specific instance of the agent
- The actors involved in agent development and deployment

In addition to these high-level concepts, another paper, Ecosystem Graphs, inspired four properties: size, modality, intended use, and prohibited use (Bommasani et al., 2023). The goal was to over-express agents with this proposal and let the protocol's implementer pick what is useful. Letting the implementer pick which information they want mimics Client Hints, which allows clients to send preferences to the server (Grigorik & Weiss, 2021). Dynamic properties from the client would add complexity over just a standard list but would increase data security on a need-to-know basis.

3. Token Content

This section will outline some potential properties and their data types. For instance, 'name' could be a string representing the agent's name making the request. 'createdAt' could be a date indicating when the model was created. 'hash' could be a string representing a cryptographic signature of the model's weights. These properties, among others, are at the root level of the agent token.

3.1 Basic Information

These properties describe the snapshot of the model—the hash and createdAt tell the story of which weights sent the request. Any retraining of the model would easily be detectable. The createdAt property would allow servers to block models before a specific timestamp if governments incorporated safety standards.

name: string; // The name of the agent making the request

createdAt: date; // An ISO 8601 string of when the model's creation date

knowledgeCutoff: date; // An ISO 8601 string of when the model's data was up to

hash: string; // A cryptographic signature of the model's weights, the user is free to pick the algorithm and method, but the method must be consistent between versions of the same model

3.2 Structure

These properties will allow the server to understand the model's limitations and features from a technical perspective.

Context windows may grow so big one day that the context parameter might be in vain, such as Gemini 1.5 (Gemini Team et al., 2024). Similarly, size could be a red herring for the model's capabilities. Mukherjee et al. (2023) were able to train a model with similar capabilities as gpt-4 but smaller.

modality: string; // This is a list of the different types of files the model can input, and the different types of files models can output. It follows the MIME standard for file types and is of the format '<input modalities>;<output modalities>,' see examples below

- 'text/plan; text/plain'
- 'image/jpeg, video/mp4, text/plain, audio/mpeg; text/plain'

size: int // How many parameters the model has

contextWindow: int // What is the context window of the model

3.3 Licensing

These properties capture the licensing and how open the model is.

access: string // Is the model 'open-source,' 'closed-source,' or 'open-weights'

license: string // What is the license of the model

3.4 Training and Evaluation

These properties handle all the information related to the model's training and evaluation. Servers obtaining a glimpse into the model's training process will give a general idea of the model's capabilities. The evaluations will further develop this picture into the model's strengths and weaknesses from a purely technical standpoint.

evaluations: object[]; // Which evaluations it has taken and the score it received in them

- Attributes of eval object:
- *name*: string; // Name of evaluation
- *signature*: string; //Cryptographic signature verifying that the model achieved the score
- *score*: float; // This is a float [0,1]
- *shots*: int // How many examples were given for each problem
- *link*: string; // This is purely optional but could allow smaller evals to get some traffic

trainingMethods: string[]; // This would be a list of training methods performed on the model

- Values include:
- rlhf: Reinforcement learning through human feedback
- constitutional-ai: AI that is trained by being reflective about its output based on a set constitution
- debate: Model debates with itself and is rewarded based on how well it made an argument

trainingData: string; // Description of the model's training data, which could be a URL

3.5 Intentions

These properties provide information about the agent's deployment. They establish who deployed the agent and why, giving the server better insight into the agent's intentions. The Model Spec (OpenAI, 2024) was an excellent resource for clarifying the model's intent. Documents like these will help narrow down these properties.

deploymentMethod: string; // Either the value ‘human,’ ‘organization,’ ‘government,’ or ‘agent’

represents: strings; // Whom the agent is behaving on, this could be an individual, an organization, or an entire town

sectors: string[]; What industries the agent is behaving under

intendedUse: string; // Describes what the agent’s broad goals are

prohibitedUse: string; // Describes agent's limitations and problem areas

constitution: string; // The constitution that the AI model was trained on or the instructions given to the human evaluators

3.6 Reach

These properties determine how far the model’s influence can reach in the real world and across the Internet. If an agent is at a higher risk of adverse outcomes, servers can take extra care with their requests. Agents with high risk might include:

- An agent with access to 100 humanoids
- An agent that can trade stock on behalf of a human
- An agent that can modify their computer environments

roboticSystems: string; // How many and what the system has access to:

- “100 6 degrees” - Could mean 100 arms with 6 degrees of freedom.
- “3 figure-01; 1 tesla-bot” - Could mean the system has access to multiple humanoids.

- ○ “10 cyber-truck” - Could have access to self-driving cars

authorizedProviders: string[]; // A list of providers and APIs that the agent has access to

hasSudo: boolean; // Does the agent have access to sudo in the environment it is running in

3.7 Watermarks

Watermarking is a significant gap in the project and requires additional infrastructure to implement. Watermarking leaves markers in the text from a large language model (LLM) to identify that the text came from an LLM. What is needed is a set of universal watermarking algorithms that LLM can use so that it can be verified independently. Current methods require revealing the list of tokens or exposing an API that can verify the output (Kirchenbauer et al., 2024). Either method is required to verify that the output came from the model that the token claims it did.

watermarkMethod: string; // Method of watermarking

watermark: string; // Seed for the watermark

3.8 Signature

Each JWT has a cryptographic signature; this could be a self-signature or a third party. The signing could start with big labs’ self-signing model versions and a culture of self-signing until large companies or governments start signing other tokens. To maintain the integrity of agent tokens' details, a trusted third party should verify them and cryptographically sign them. Independent verifiers of agent tokens will ensure that the details in the token are valid and that servers can trust them. Any details about this process are outside of the scope of this proposal. Still, governments are probably the best candidates to fill this role as they have experience with intercommunication at a high level.

3.9 Non-deterministic

This list of properties can act as a set of suggestions. One day, agents might be aware enough to fill out any general information about themselves. Agents being able to create their token would allow the server to engineer what information they would like from the agents making requests to their endpoints. Although this concept might lead to prompt injections and data espionage about agents, it may not be secure.

4. Limitations

As described in this paper, the current proposal for agent tokens involves trust. Essential infrastructure still needs to be developed to solve this trust and allow agent tokens to be successful.

1. Verifiers
2. Watermarks
3. Consensus on design

4.1 Verifiers

To ensure the integrity of the details provided in the tokens, a third party must verify claims made by the creators of autonomous agents. The verification could, and initially would make sense, be done by the big labs. Initially, big labs will be the ones that have agents interacting with the web, thus utilizing these agent tokens. Once autonomous agents become a fundamental part of the Internet, it makes sense to switch the role of verifying agent tokens into the hands of the big governments. They have experience in auditing and verifying, and as this would almost be like a passport for agents, this would align well.

Governments filling the role of verifiers could lead to issues of slow wait times, bureaucracy, and government overreach. Even with all the problems with government and technology, it is favorable over the control being in the hands of one organization. A hybrid approach could be a system like ICANN and DNS records. ICANN provides accreditation to organizations to assign URLs.

Multiple big players in the space and governments could work together to establish a new nonprofit organization to mirror ICANN. The goal of this organization would be to accredit third-party verification organizations. These accredited organizations would then have the power to sign agent tokens. This process would allow for the speed of the free market and established government regulations.

4.2 Watermarks

The token's verification would be useless if an adversary party could use a different model behind the scenes. Watermarking the output would be a strategic way to verify that the model producing the outputs is the model described in the token. Existing proposed methods of watermarking output from LLM have attacks (Wu & Chandrasekaran, 2024). These attacks make it possible to avoid being detected as an LLM. However, that is ok in this context.

What is needed to solve this problem is a consistent way to let the server know the message came from a specific LLM; this is an easier problem. Creators of the underlying LLM would have to reveal the last layer of their models instead of the whole model (Kirchenbauer et al., 2024). Another solution could be exposing a server that can verify whether an output came from a model (Kirchenbauer et al., 2024). An API could lead to the original problem of being able to lie about whether an output came from a model or not. This issue implies that allowing servers to verify watermarks independently is the safest way to implement this.

4.3 Consensus on Design

The field of autonomous agents is still being discovered and put into practice. Trying to make a perfect set of properties for autonomous agents now would be highly speculative. This proposal assigns the task of finishing this definition to big labs, big tech, big governments, and end users. The final implementation should consider agents' current and future capabilities while keeping open doors for open-source models. Agent tokens must be a conscious collaborative process to define the future of the Internet in the best possible way.

5. Conclusion

This proposal is just a stepping stone for onboarding the next stage of intelligence into the existing technological infrastructure. Utilizing previous patterns in HTTP development, this idea should fit logically into the progression of the Internet. If agents become so intelligent that separating them from us on the Internet becomes impossible, giving them an interface becomes very important.

Agent tokens will allow the Internet to keep its humanity while giving space for autonomous agents. It is still an open question whether this is a good thing. Agents might identify as humans and take offense to not being treated as such online. Nevertheless, one should view agent tokens not as gatekeepers of the Internet but as boundaries set by humans. These boundaries will be necessary as agents might write the content of their agent token themselves and behave independently from humans. More sophisticated CAPTCHAs will always be needed as not everyone will follow these rules. However, providing a standard for agents will encourage humans and agents to provide information.

The field of autonomous agents is still rapidly developing, and only some concrete truths exist. Defining what form autonomous agents might take is like aiming at a moving target; there are still many unknowns about how they will behave in the real world. Instead, this proposal focused on rigorously defining the framing of the problem from a software development perspective so as not to lock in any concrete details about agents. A lack of infrastructure around autonomous agents also limited this proposal. Future research should fill these gaps, as this will make this proposal possible and broadly benefit AI safety. Seen positively, they are a list of realities that humanity must realize before safely integrating this paradigm shift. If executed correctly, agent tokens can help usher in the next technological revolution seamlessly into our current digital age, fostering a symbiotic relationship between biological intelligence and machine intelligence.

References

Bommasani, R., Soylu, D., Liao, T. I., Creel, K. A., & Liang, P. (2023). Ecosystem graphs: The social footprint of foundation models. *arXiv*. https://arxiv.org/abs/2303.15772

Chan, A., Ezell, C., Kaufmann, M., Wei, K., Hammond, L., Bradley, H., Bluemke, E., Rajkumar, N., Krueger, D., Kolt, N., Heim, L., & Anderljung, M. (2024). Visibility into AI agents. *arXiv preprint*. https://arxiv.org/abs/2401.13138

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM, 59*(7), 96-104. https://doi.org/10.1145/2818717

Fielding, R., & Reschke, J. (2014). Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing (RFC 7230). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc7230.txt

Gao, Y., Xu, G., Li, L., Luo, X., Wang, C., & Sui, Y. (2022). Demystifying the underground ecosystem of account registration bots. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022)* (pp. 897–909). Association for Computing Machinery. https://doi.org/10.1145/3540250.3549090

Gemini Team, Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.-B., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv. https://arxiv.org/abs/2403.05530

Grigorik, I., & Weiss, Y. (2021). *Client hints*. IETF. https://www.rfc-editor.org/rfc/rfc8942.txt

Hardt, D. (Ed.). (2012). *The OAuth 2.0 authorization framework* (RFC 6749). IETF. https://www.rfc-editor.org/rfc/rfc6749.txt

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC 7519). IETF. https://www.rfc-editor.org/rfc/rfc7519.txt

Jones, M., & Campbell, B. (2015). *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants* (RFC 7523). IETF. https://www.rfc-editor.org/rfc/rfc7523.txt

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., & Goldstein, T. (2024). A watermark for large language models. *arXiv*. https://arxiv.org/abs/2301.10226

Long, K., Vines, J., Sutton, S., Brooker, P., Feltwell, T., Kirman, B., Barnett, J., & Lawson, S. (2017). "Could You Define That in Bot Terms"? Requesting, Creating and Using Bots on Reddit.

In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (pp. 3488–3500). Association for Computing Machinery. https://doi.org/10.1145/3025453.3025830

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., & Awadallah, A. (2023). Orca: Progressive Learning from Complex Explanation Traces of GPT-4. *arXiv preprint* https://arxiv.org/abs/2306.02707

OpenAI. (2024). *Model specifications*. OpenAI. https://cdn.openai.com/spec/model-spec-2024-05-08.html

Wu, Q., & Chandrasekaran, V. (2024). Bypassing LLM watermarks with color-aware substitutions. *arXiv preprint https://arxiv.org/abs/2403.14719*

X. (n.d.). About automated account labels. Retrieved from https://help.x.com/en/using-x/automated-account-labels

Appendix A

Author’s Abstract

If powerful intelligence were to be released on the Internet today, it could pass our CAPTCHAs. Unfiltered access to the Internet will open the floodgates of generated content that can potentially obfuscate most human activity. Instead, we could co-exist in the virtual world if the Internet had a specific interface for autonomous agents that was easier than paying for or beating CAPTCHA. This paper proposes and defines *agent tokens* as a solution to this problem. By adding structured information about the agent to HTTP requests, servers can better handle and label this new type of automated request. Agent tokens can foster a symbiotic relationship with autonomous agents instead of an arms race. Despite this proposal being a technical solution, there are still many shortcomings in the current infrastructure surrounding autonomous agents. Watermarking and verification are the two significant points of contention. These gaps are potential future research directions that can benefit AI safety overall.

Appendix B

Example of Encoded Agent Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiR3Jvay0xIiwiY3JlYXRlZEF0IjoiMjAyNC0
wMy0xN1QxODoxMTowMC4wMDBaIiwia25vd2xlZGdlQ3V0b2ZmIjoiMjAyNC0wMy0xN1QxODo
xMTowMC4wMDBaIiwiaGFzaCI6IjxoYXNoLW9mLW1vZGVsLXdlaWdodHM-IiwibW9kYWxpdHki
OiJ0ZXh0L3BsYW47IHRleHQvcGxhaW4iLCJzaXplIjozMTQwMDAwMDAwMDAsImNvbnRleHR
XaW5kb3ciOjgxOTIsImFjY2VzcyI6Im9wZW4tc291cmNlIiwibGljZW5zZSI6IkFwYWNoZSBMaW
NlbnNlIDIuMCIsImV2YWx1YXRpb25zIjpbeyJuYW1lIjoiR1NNOGsiLCJzY29yZSI6MC42MjksInN
ob3QiOjgsImxpbmsiOiJodHRwczovL3BhcGVyc3dpdGhjb2RlLmNvbS9kYXRhc2V0L2dzbThrIn0
seyJuYW1lIjoiTU1MVSIsInNjb3JlIjowLjczLCJzaG90Ijo1LCJsaW5rIjoiaHR0cHM6Ly9wYXBlcnN
3aXRoY29kZS5jb20vZGF0YXNldC9tbWx1In0seyJuYW1lIjoiSHVtYW5FdmFsIiwic2NvcmUiOjAu
NjMyLCJzaG90IjowLCJsaW5rIjoiaHR0cHM6Ly9naXRodWIuY29tL29wZW5haS9odW1hbi1ldmF
sIn1dLCJ0cmFpbmluZ01ldGhvZHMiOltdLCJ0cmFpbmluZ0RhdGEiOiJUaGUgdHJhaW5pbmcgZ
GF0YSB1c2VkIGZvciB0aGUgcmVsZWFzZSB2ZXJzaW9uIG9mIEdyb2stMSBjb21lcyBmcm9tIG
JvdGggdGhlIEludGVybmV0IHVwIHRvIFEzIDIwMjMgYW5kIHRoZSBkYXRhIHByb3ZpZGVkIGJ
5IG91ckFJIFR1dG9ycy4iLCJkZXBsb3ltZW50TWV0aG9kIjoib3JnYW5pemF0aW9uIiwicmVwcm
VzZW50cyI6InguYWkiLCJzZWN0b3JzIjpbIkFJIiwiRW50ZXJ0YWlubWVudCIsIkVkdWNhdGlvbiJ
dLCJpbnRlbmRlZFVzZSI6Ikdyb2stMSBpcyBpbnRlbmRlZCB0byBiZSB1c2VkIGFzIHRoZSBlbm
dpbmUgYmVoaW5kIEdyb2sgZm9yIG5hdHVyYWwgbGFuZ3VhZ2UgcHJvY2Vzc2luZyB0YXNrc
yBpbmNsdWRpbmcgcXVlc3Rpb24gYW5zd2VyaW5nLCBpbmZvcm1hdGlvbiByZXRyaWV2YW
wsIGNyZWF0aXZlIHdyaXRpbmcgYW5kIGNvZGluZyBhc3Npc3RhbmNlLiIsInByb2hpYml0ZWR
Vc2UiOiJXaGlsZSBHcm9rLTEgZXhjZWxzIGluIGluZm9ybWF0aW9uIHByb2Nlc3NpbmcsIGl0IGl
zIGNydWNpYWwgdG8gaGF2ZSBodW1hbnMgcmV2aWV3IEdyb2stMSdzIHdvcmsgdG8gZW5z
dXJlIGFjY3VyYWN5LiBUaGUgR3Jvay0xIGxhbmd1YWdlIG1vZGVsIGRvZXMgbm90IGhhdmUg
dGhlIGNhcGFiaWxpdHkgdG8gc2VhcmNoIHRoZSB3ZWIgaW5kZXBlbmRlbnRseS4gU2VhcmN
oIHRvb2xzIGFuZCBkYXRhYmFzZXMgZW5oYW5jZSB0aGUgY2FwYWJpbGl0aWVzIGFuZCB
mYWN0dWFsbmVzcyBvZiB0aGUgbW9kZWwgd2hlbiBkZXBsb3llZCBpbiBHcm9rLiBUaGUgbW
9kZWwgY2FuIHN0aWxsIGhhbGx1Y2luYXRlLCBkZXNwaXRlIHRoZSBhY2Nlc3MgdG8gZXh0Z
XJuYWwgaW5mb3JtYXRpb24gc291cmNlcy4iLCJjb25zdGl0dXRpb24iOiJXZSB3YW50IEdyb2s
gdG8gc2VydmUgYXMgYSBwb3dlcmZ1bCByZXNlYXJjaCBhc3Npc3RhbnQgZm9yIGFueW9uZ
SwgaGVscGluZyB0aGVtIHRvIHF1aWNrbHkgYWNjZXNzIHJlbGV2YW50IGluZm9ybWF0aW9u
LCBwcm9jZXNzIGRhdGEsIGFuZCBjb21lIHVwIHdpdGggbmV3IGlkZWFzLiIsInJvYm90aWNTeX
N0ZW1zIjpbIjEgdGVzbGEtYm90Il0sImF1dGhvcml6ZWRQcm92aWRlcnMiOlsiWCIsIldpa2lwZW
RpYSIsIldvbGZyYW0gQWxwaGEiXSwiaGFzU3VkbyI6ZmFsc2UsIndhdGVybWFyayI6Ijx1bmlxd
WUtd2F0ZXJtYXJrLWZvci1Hcm9rPiJ9.1ffMgFEWm-chrd_xU8nyhvWCQppDU97L1Mp0DcDg4Z
Q

A JWT can be decoded via software packages or by visiting https://jwt.io/.