Towards Anonymous Chatbots with (Un)Trustworthy Browser Proxies

Dzung Pham, Jade Sheffey, Chau Minh Pham, Amir Houmansadr
University of Massachusetts Amherst
{dzungpham, jsheffey, ctpham, amir}@cs.umass.edu

***Abstract*—AI-powered chatbots (ChatGPT, Claude, etc.) require users to create an account using their email and phone number, thereby linking their personally identifiable information to their conversational data and usage patterns. As these chatbots are increasingly being used for tasks involving sensitive information, privacy concerns have been raised about how chatbot providers handle user data. To tackle this issue, we designed and built ProxyGPT, a privacy-enhancing system that enables anonymous queries in popular chatbot platforms. ProxyGPT leverages volunteer proxies to submit user queries on their behalf, thus providing network-level anonymity for chatbot users. The system is designed to support key security properties such as content integrity via TLS-backed data provenance and end-to-end encryption while also ensuring usability and sustainability. To the best of our knowledge, ProxyGPT is the first comprehensive proxy-based solution for privacy-preserving AI chatbots. Our codebase is publicly available at `https://github.com/dzungvpham/proxygpt`.**

I. Introduction

The introduction of AI chatbots, including ChatGPT (OpenAI), Claude (Anthropic), and Gemini (Google) [2, 23, 52], has significantly transformed how people interact with artificial intelligence (AI) technologies. Powered by large language models (LLMs) that are pre-trained on massive amounts of data, these chatbots can perform various complex tasks previously thought exclusive to humans, ranging from basic functions such as writing and programming assistance [38, 65] to more complex and sensitive use cases like providing medical, legal, or financial advice [11, 36, 37].

As chatbots become increasingly integrated into our daily lives, privacy concerns regarding sensitive user data have emerged [43]. Many popular chatbot platforms require users to create verified accounts using email addresses or phone numbers, effectively linking user identities to their platform activities (Table I). This practice not only prevents truly private conversations but also poses substantial privacy risks, as the information extracted from these identity-linked chats can be used to train LLMs or enable targeted advertising. ChatGPT, for instance, includes a cross-chat memory feature where machine learning algorithms extract information from different chat conversations to personalize future interactions [54]. While some chatbots like Perplexity [60] allow anonymous use of their services, users still need to sign up with their personal information to access full, unlimited functionality.

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2025
28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-9-0
https://dx.doi.org/10.14722/madweb.2025.23005
www.ndss-symposium.org

TABLE I. Identity verification requirements by some major LLM chatbots (as of December 2024). Traffic data between March and May 2024 is from semrush.com.

Chatbot	LLM family	Identity required?	Estimated monthly visits
ChatGPT [52]	GPT	Yes [a,b]	2.7B
Gemini (Bard) [23]	Gemini	Yes	170.8M
Claude [2]	Claude	Yes	57.5M
Copilot [41]	GPT	Yes	43.4M
Meta AI [40]	Llama	Yes	5.2M
Grok [75]	Grok	Yes	1.8M
Perplexity [60]	Mixed	No [b]	58.0M
YouChat [76]	GPT	No [b]	12.0M
HuggingChat [28]	Mixed	No [b]	2.4M

[a] ChatGPT recently no longer requires users to sign in [55], but we find this mode to be unavailable for Tor and certain VPN locations.
[b] Requires signing in to access full functionalities.

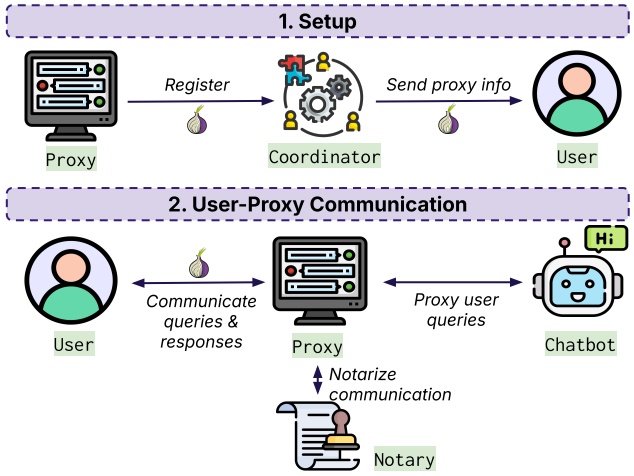

Fig. 1. Overview of ProxyGPT. Users select volunteer proxies via the coordinator. Proxies are randomly audited using a TLS data provenance protocol. All communications (except between proxies and chatbots) utilize AC protocols like Tor.

We argue that users should have the option for *anonymized access* to chatbots specifically and to AI technologies broadly for two key reasons. First, anonymizing chatbot usage can enhance user privacy by decoupling sensitive conversational data from user identities. Second, anonymity can promote trust in AI by enabling people to freely express ideas and opinions without fear of retaliation or surveillance from chatbot service providers. Therefore, this work tackles the following question:

How can we design and implement a system that enables users to access chatbots anonymously? We consider the following properties to be essential to our design:

- *User Anonymity*: Chatbot providers and other system participants should not be able to determine the true identity of the query authors. This is our most important goal.
- *Content Integrity:* Privacy providers should not be able to tamper with user queries or respond with false information. Dishonest providers should be identified and restricted.
- *Least Privilege*: The content of the queries and responses should only be visible to relevant parties.
- *Usability:* Users and privacy providers should be able to participate in our system with minimal setup and without having to pay real money.
- *Sustainability:* The system should offer incentives for privacy providers to participate and should scale with the number of users and privacy providers.
- *Resistance to Sybil and Denial-of-Service (DoS):* The system should have mechanisms to prevent users and proxies from gaining an oversized influence and to mitigate DoS attacks.

Achieving all of these objectives without any assistance from service providers is challenging for high-level web applications like chatbots. Naively using a temporary or shared account to hide one's identity (e.g., using services like TeroBox [71]) can run into many obstacles such as device fingerprint leakage and usage rate-limiting. To address these challenges, we design and implement ProxyGPT, a privacy-preserving chatbot solution that takes a user's query and securely sends it to volunteer proxies who interact with chatbots on behalf of the user (Figure 1). Our system includes a *coordinator* that manages a directory of volunteer *browser-based proxies* and facilitates user-proxy communications backed by an anonymous communication (AC) protocol such as Tor [16]. The query and response content is end-to-end encrypted and accessible to authenticated parties only. To prevent proxy volunteers from tampering with the chatbot requests and responses, we integrate into ProxyGPT a *TLS-backed data provenance* protocol [77] called TLSNotary [32], which generates cryptographically verifiable proofs for the authenticity of proxy responses. Proxies must regularly complete integrity audits with TLSNotary to participate in our system. Due to the high computational overhead of TLSNotary, we also incorporate a simple reputation system where users can rate proxies to further enhance trustworthiness.

ProxyGPT's capacity and sustainability depend on the number of proxy volunteers. The more proxies there are, the more users we can serve and the more resilient our system is to any disruptions. To incentivize proxy volunteers, we implement a lightweight anonymous payment scheme based on Chaum's blind-signature electronic cash [7]. Proxies can obtain this e-cash simply by completing our automatic integrity audits. The e-cash can then be spent within ProxyGPT to ask more queries without linking the spender to the original proxy who obtained the e-cash. The use of integrity audits and e-cash also serve as a DoS and Sybil mitigation.

To demonstrate the usability of ProxyGPT, we implement a fully functional proof-of-concept consisting of two major components. The first is a coordinator deployed as a Tor hidden service, featuring a modern web interface for users to securely interact with proxies. Users can enjoy private chat experiences with different chatbots like ChatGPT and Claude. The second component is a browser extension for Chromium-based browsers that allows anyone to act as a proxy. Our performance evaluation shows that ProxyGPT users receive responses to their queries within 15 seconds on average. Proxy volunteers, however, may take up to two minutes to complete a single integrity audit, depending on networking conditions. We additionally provide an extensive discussion of the various privacy, security, integrity, and ethical aspects of our system.

To the best of our knowledge, our work is the first to offer a comprehensive proxy-based privacy-enhancing solution for AI chatbot users, fully considering crucial aspects like data provenance and sustainability. In fact, we believe our system is also the first application to combine proxying and TLS data provenance for high-level web data. We envision ProxyGPT as a platform that can augment the capability and popularity of chatbot providers rather than circumventing them, thus bringing their services to an even broader audience who might feel hesitant due to privacy concerns.

II. Background & Related Work

In this section, we provide a brief overview of the current state of user privacy in LLM chatbots and privacy-preserving communication techniques, along with an overview of non-repudiation for TLS-protected data.

A. User Privacy in LLM Chatbots

The privacy of LLM technology has been increasingly under scrutiny from academia, industry, and the general public, with various privacy risks identified in the LLMs such as training data memorization [4, 48, 50], sensitive data inference [43, 69], and unsatisfactory privacy reasoning [44]. Several works have attempted to address privacy leakage in text data, particularly user's chatbot prompts, by using LLMs themselves to perform text sanitization [10, 63, 68, 70, 79]. Unlike these, ProxyGPT aims to hide user identities at the application level (i.e. emails and phone numbers) and network level (i.e. IP addresses). The most similar work to ours is AnonChatGPT [1], which claims to allow users to interact anonymously with OpenAI's ChatGPT. The service provides a regular web page that comes with various advertisements as well as Google Analytics tracking scripts. It does not appear to use the same web version of ChatGPT due to the difference in response length and quality, and it does not support multi-query conversations, resulting in limited utility to users. Additionally, the service is not open-source and has no formal guarantee that the responses truly come from OpenAI's ChatGPT. We believe that the service is a single server-based proxy relying on OpenAI's API which costs real money to use (hence the ads). While AnonChatGPT's intention is valid, we believe it has much room for improvement with regard to user privacy and trust. Stronger privacy-preserving inference alternatives such as homomorphic encryption [8] cannot yet support LLMs with hundreds of billions of parameters.

B. Anonymous Communications

Various anonymous communication (AC) protocols have emerged over the last two decades to address the increasing need for internet privacy [66, 67]. Most notable is The Onion Router project (Tor), which features a decentralized network of volunteer-based relays that proxy internet traffic for users through a series of nodes while hiding the sender/receiver information behind layers of encryption [16]. Unfortunately, when applied to the problem of online LLM chatbots, AC systems like Tor do not automatically grant users any additional privacy since many current LLM chatbots still require users to sign in to identity-verified accounts even for their free service. As a result, user identities are still transmitted in the clear to the chatbot providers even when AC is utilized.

Closely related to our system are peer-based censorship circumvention applications like Snowflake [3], MassBrowser [49], and Hola VPN [27]. These systems rely extensively on browser-based volunteers who help proxy traffic for other users. This approach is also adopted in our ProxyGPT design, but instead of proxying arbitrary internet traffic, we focus specifically on interactions with chatbots. The idea of using proxies to enable anonymous access to web services has also been explored in the early days of the web with systems like Janus [22], Lucent [21], and UPIR [17], but they do not consider or support untrustworthy peer-based proxies.

C. Provenance of TLS-protected Data

Communication over the Internet is widely protected with the Transport Layer Security (TLS) protocol [64] which enables a secure channel between clients and servers. TLS by itself, however, does not allow users to prove the origin and authenticity of their data to third parties since the servers are not required to sign the data, and the users can forge the data using the symmetric TLS session key shared with the server. Proving the provenance of TLS-protected data is essential to promoting trust in our ProxyGPT system as we can provide users with formal guarantees that the responses they receive from volunteer proxies indeed originate from the relevant chatbot systems with the appropriate query context (e.g., correct queries, chatbot model types, decoding strategy, etc.). Recent work such as DECO [77] and TLSNotary[1] [32] has demonstrated the feasibility of TLS-based non-repudiation via a novel protocol involving a neutral verifier, secure multiparty computation (MPC), and zero-knowledge proofs (ZKP). At a high level, the protocol consists of three phases: First, the user and the verifier jointly negotiate the TLS session key with the server via a three-way handshake protocol to each obtain a secret key share. Then, the user performs MPC with the verifier to encrypt the user's message without revealing the plaintext to the verifier and sends it to the server. Finally, after committing the TLS session data to the verifier, the user obtains the verifier's secret key share and generates (zero-knowledge) proof about the data via selective redaction/revelation. While this approach does not require any cooperation from the data server and can keep the data private from even the verifier, it has a high overhead when executed over the Internet due to the MPC operations (Table III). Consequently, we cannot require this protocol for every single proxy request in ProxyGPT and must take a more economical approach.

[1]TLSNotary was previously known as PageSigner, but has been recently rewritten from scratch and modeled after the DECO protocol.

III. ProxyGPT Participants and Threat Models

We present a high-level overview and threat model of the participants in ProxyGPT, including chatbot providers, users seeking query anonymity, and volunteer proxies (Figure 1). We assume that these actors are malicious adversaries that can take active actions to undermine the privacy service, uncover the identities of query owners, or disrupt the functionality of ProxyGPT.

A. Chatbot Providers

We refer to entities that provide LLM-powered chatbot services as *chatbot providers* (e.g., OpenAI, Google, Anthropic). Our threat analysis of chatbot providers is based on their business models and usage policies. While they may not yet operate at the level of nations or states, chatbot providers are fully capable of monitoring user activities on their platforms, controlling and mining any permitted generated content, and limiting user access to their services.

1) Usage: Chatbot providers can offer users two methods for interacting with their services: a graphical user interface (GUI) and an application programming interface (API). The former method typically includes a free tier with basic functionalities and a paid tier with more features. The latter is usually charged per the number of input and output tokens used. Payment methods are often restricted to traditional electronic payment systems (e.g., debit/credit cards), and support for (pseudo)anonymous payment schemes like cryptocurrencies is still limited.

Chatbot providers can require users to sign up for an account with their e-mail and/or other sensitive personal data such as phone numbers before they are allowed access to the chatbot services (Table I). This private information is typically used for security purposes such as human verification or account recovery. To use the GUI, users often need to log in to their accounts and may periodically be asked to complete bot prevention challenges. To use the API, users need to use sufficiently funded unique API keys.

2) Mining User Conversation: Content generated on the chatbot platforms can be incorporated into the training of the chatbots, which are known to be prone to verbatim memorization [4] and training data leakage [48]. This content can also be analyzed to create a profile of the user to provide a more personalized chat experience such as customized chatbot responses or recommendations [54]. Previous research has demonstrated that a great deal of sensitive information can be inferred from user web search logs [31] and chatbot prompts [43, 69]. Users may be offered the ability to opt out of these features, but this depends on the chatbot services.

3) Modifying Chatbot Responses: Chatbot providers can easily modify the chatbot responses. For instance, they can inject into the chatbot responses special watermarks that are not easily detectable by humans but can be identified algorithmically [35]. One proposed watermark works by coercing the statistical distribution of the tokens generated by the chatbots into a uniquely distinguishable "shape". These watermarks can

hypothetically be used to link a user and their proxied chatbot responses, particularly when the users reuse the watermarked text in their regular non-proxied queries.

4) Activity Restriction: Chatbot providers can impose usage limits (e.g. number of queries per hour) as well as restrict access for any user, regardless of whether the user actually violates any usage terms or policies. In reality, we expect the chatbot providers to be *reasonable* with our proxy-based privacy approach since the act of proxying by itself does not violate any terms of service (Section VII-D), and arbitrarily restricting users without a justifiable reason can negatively impact the reputation of the chatbot providers. Other restrictions may involve dropping queries or delaying responses.

B. Proxies

Proxies are volunteers who interact with chatbots on behalf of ProxyGPT users. Therefore, a proxy needs to use its service identity to interact with chatbots and requires access to the query plaintext.

1) Dishonest Service: Since chatbot services may charge a non-negligible usage fee (e.g. for API access), proxies might be incentivized to reduce costs by modifying the original query or the API settings such as limiting the number of input/output tokens. It is also possible for proxies to provide a completely dishonest response using cheap but subpar chatbots or even random information. More maliciously, proxies can intentionally respond with inappropriate content or with instructions or code that can compromise the security and privacy of ProxyGPT users (e.g., cross-site scripting).

2) Collusion: Proxies and chatbot providers may collaborate in an attempt to identify the true owners of the queries. For example, proxies can inform chatbot providers of the queries received from users along with any relevant metadata. Chatbot providers can then attempt to find the true owners of the queries (e.g., via stylometry with previous chat logs [46]). Chatbot providers may pose as proxies themselves to directly obtain such information.

C. Users

ProxyGPT users are those who wish to use chatbot services anonymously. By design, any information related to their identity such as their chatbot accounts and IP should not be revealed to chatbot providers, proxies, or even trusted third parties (Section III-D).

1) Malicious Queries: Similar to proxies responding with bad content, users can also ask malicious or inappropriate queries with the intention of harming proxies. For example, improperly validated queries might lead to attacks such as cross-site scripting.

2) Collusion: Users may collude with chatbot providers to identify proxies. For instance, chatbot providers can pose as ProxyGPT users and ask uniquely identifiable queries. By tracking the accounts or API keys that submit these fingerprinted queries, they can reveal the proxies involved.

D. Trusted Third Parties

There are two types of trusted third parties (TTP) participating in ProxyGPT, namely the *coordinator* and the *notary*.

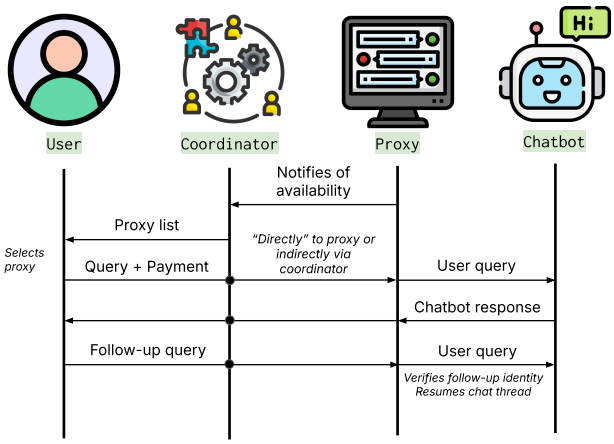

Fig. 2. Query proxying process. Users can send queries "directly" over Tor to a server-based proxy or via the coordinator to a browser-based proxy. A form of payment may need to be sent to the proxy or the coordinator.

1) Coordinator: The coordinator's main responsibilities include managing a directory of proxies, helping users discover proxies, and facilitating communication between users and proxies (similar to the "central bridge" in Snowflake [3] and the "operator" in MassBrowser [49]). Both users and proxies only need to trust that the coordinator faithfully follows ProxyGPT's protocol without having to share the content of their chatbot queries or any identity-linked information like IP. Any new proxy must register with the coordinator and complete a verification process before it can participate in ProxyGPT. Users can contact the coordinator for a list of proxies and establish either a direct or a coordinator-facilitated communication channel with chosen proxies. To promote trustworthiness, the coordinator performs regular audits with the help of the *notary*.

2) Notary: The notary's primary job is to verify and notarize the TLS-based communication session between a proxy and a chatbot service using a protocol like DECO [77] or TLSNotary [62]. The notarized session can then be independently verified by other parties such as the coordinator or the users themselves. By design, the information visible to the notary/verifier only includes the server's hostname and any session data that has been selectively revealed by the proxies. As such, the identities of proxies can remain hidden, while the text content of the chatbot conversation will need to be revealed for verification purposes. We assume that the notary does not collude with proxies to produce inauthentic notarization.

IV. System Operation Details

In this section, we describe two important operations in ProxyGPT: submitting a query (Figure 2) and verifying a proxy (Figure 3).

A. Query Submission

1) Proxy info request: A user who wants to use ProxyGPT first anonymously requests a list of available proxies from the coordinator. The coordinator retrieves its directory of proxies, filters for those that are recently active (e.g. last contact within

the last five minutes), and then replies to the user with the proxies' contact information such as their onion addresses or ProxyGPT pseudonyms along with some performance statistics to help users choose.

2) Proxy selection: Users are provided with various proxy performance statistics calculated periodically by the coordinator to aid the proxy selection process. These include mean response time, average daily request volume, and downvote rate (Section V-B6).

3) Communication with proxy: Using the selected proxy's contact information, users can establish a "direct" connection with a chosen proxy via an AC protocol like Tor. If the connection succeeds, users can proceed to send their queries to the proxy along with a form of payment. If users cannot directly connect to the chosen proxy, such as in the case of browser proxies, they can instead communicate indirectly over the coordinator, which acts as a dropbox for the queries. To prevent the coordinator from seeing the query content, the query content is end-to-end encrypted.

4) Rating proxy response: Since a proxy may fabricate its response (Section III-B1), users should ideally be able to request the proxy to provide a formal cryptographic proof of the response's authenticity via a protocol like DECO or TLSNotary. However, due to the high communication overhead of these protocols, we limit the proof requirement to audit requests from the coordinator only. Users can instead provide a rating of the proxy responses, specifically by downvoting bad responses, which will impact the reputation of the proxies. (Alternatively, users can perform a manual cross-check by asking the same query to other proxies and comparing the results, but this is more costly.)

5) Payment: Certain proxies may require users to pay for their services to offset the operational costs. OpenAI's GPT, for example, imposes a fee for every input and output token for its API. To preserve privacy, users can pay with cryptocurrencies, such as those with low transaction fees. However, we recognize that requiring users to pay with real money will likely hinder the adoption of ProxyGPT. As such, we also include an alternative payment method where users can "pay" by simply volunteering to proxy for others. For every successfully validated verification request, proxies can obtain from the coordinator a "certificate" that can be used to pay for ProxyGPT. This payment model applies to browser-based proxies who can make free requests to chatbots via a web-based UI. To avoid associating a proxy's certificate with their own queries, we use Chaum's blind-signature-based electronic cash scheme [6, 7] to allow the proxy to privately use their certificate without anyone learning the identity of the certificate's owner, even the coordinator.

B. Proxy Registration & Audit

A proxy must register itself with the coordinator and complete an integrity audit before it can participate in ProxyGPT:

- *Registration request*: The proxy sends the coordinator a unique 'pseudonym' for communication purposes (e.g., onion address in Tor or a public key), along with a list of constraints such as the supported chatbots.
- *Verification challenge*: Upon receiving a registration request, the coordinator prepares and sends the proxy a verification challenge (Figure 7). The challenge should also be answered within a reasonable time frame (e.g., 10 minutes).
- *Verification response*: The proxy first obtains the responses from the target chatbots. To prove authenticity, the proxy then participates in the TLSNotary protocol with a notary. After that, it sends the responses along with the proof to the coordinator. The proof should hide any sensitive information that may reveal the proxy's identity (Figure 8).
- *Registration notification*: The coordinator verifies the proxy's responses and proof and notifies the proxy of the result. If the proxy fails, it must restart the registration.

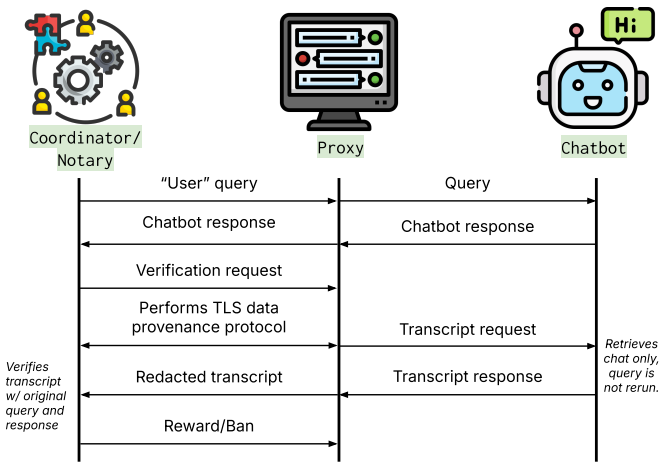

Fig. 3. Proxy auditing. The coordinator sends the proxy a challenge query disguised as a normal one. After obtaining the proxy's response, the coordinator requests the proxy to participate in a TLS data provenance protocol like DECO to ensure the original query and response are authentic.

The registration process thus serves two purposes: to ensure that proxies are capable of correctly performing data provenance protocols and to mitigate DoS attacks. However, a registered proxy can still mishandle user requests later on. To prevent this, the coordinator also performs regular random audits of the proxies (Figure 3) by sending them special challenge queries specifically generated to prevent proxies from detecting when an audit is happening (Section V-A3). A proxy with pending audit queries will no longer receive new genuine user requests until the audit queries are processed. After the proxy sends the coordinator the chatbot responses to the audit queries, the coordinator then requires the proxy to provide a TLSNotary proof of the responses. The proxy only needs to make a request to the chatbot providers to retrieve the relevant conversation for the proof without having to redo the audit queries. A proxy that fails this process will be banned from ProxyGPT and will have to re-register. A proxy who succeeds will be rewarded with our e-cash that can be used to ask queries with ProxyGPT.

V. Implementation Details

This section presents various implementation choices for our ProxyGPT proof-of-concept. We open-source our code at `https://github.com/dzungvpham/proxygpt`.

A. Coordinator & Notary

1) Deployment: We deploy the coordinator as a Tor hidden service [16] on an Ubuntu virtual machine and implement it with the Flask web framework [58], Gunicorn Python WSGI server [9], and Nginx as a reverse proxy [19]. It exposes several REST endpoints for retrieving proxy info, registering proxies, sending/receiving queries and responses, etc., and maintains a MySQL database that stores proxy information such as pseudonyms, registration status, and performance metrics, along with (E2EE) user-proxy communication data facilitated by the coordinator. All communication data is automatically deleted after 30 days.

For the notary, we deploy a Rust-based TLSNotary server (v0.1.0-alpha.5) [62] in a separate Ubuntu virtual machine along with a noVNC-based WebSocket (WS) proxy [51]. We choose TLSNotary as it is, to our knowledge, the only open-sourced, browser-friendly, and actively maintained framework that provides TLS data provenance based on zero-knowledge proof [32]. Unlike the coordinator, the notary is hosted publicly due to the prohibitive communication overhead when performing the TLS data provenance protocol over Tor. As a result, proxies need to use a faster AC system such as Mullvad VPN [45] to be able to execute TLSNotary while also protecting their privacy. The WS proxy is needed because browser-side code cannot open raw sockets required for TLSNotary. (ProxyGPT proxies can also use their own local WS proxy instead of the publicly hosted one, but this requires an extra setup step.)

2) User Interface: The coordinator provides a simple and convenient Tor website for users to discover proxies and make their proxy requests (Figure 4). The website's design is influenced by existing chatbots, particularly ChatGPT. Users can easily switch between ChatGPT and Claude, engage in multi-query conversations, as well as downvote individual responses. To preserve the styling and formatting of the original chatbot responses while also preventing cross-site scripting attacks (XSS), we use the same styling framework from ChatGPT and Claude and sanitize the responses with DOMPurify [13].

To submit a query in ProxyGPT, the client-side logic generates an Elliptic Curve Diffie-Hellman (ECDH) and an Elliptic Curve Digital Signature Algorithm (ECDSA) key pairs on curve P-256 for encrypting the query and decrypting the response end-to-end while also ensuring authenticity. The client's payload is encrypted using an AES-GCM-256 key derived from the private ECDH key and the chosen proxy's public ECDH key. The client keys are not reused across different queries to prevent them from being linked together. In addition to the query content, extra options like which chatbot to use and which thread to continue are encrypted and thus not accessible even by the coordinator. All cryptography operations use the browser-native WebCrypto API [73].

3) Proxy Audit: For every real user request to a proxy, the coordinator inserts its own query (Figure 7) disguised as a regular user request with a fixed probability (e.g., 25%) so that on average, each proxy would receive 4-6 fake queries before hitting the chatbot providers' hourly rate limit. Each of these fake queries in turn has a 50% chance of turning into a TLSNotary verification challenge. This *data poisoning* strategy is designed to confuse the proxies and prevent them from being

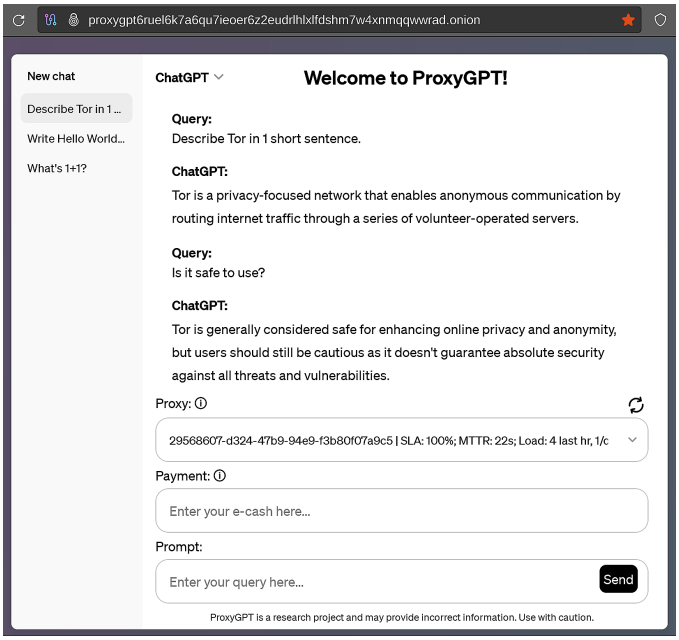

Fig. 4. User interface of our ProxyGPT website (screenshot taken from within the Tor browser). Users can switch between chatbots, select which proxy to use, engage in multi-query conversations, etc.

able to distinguish between regular and audit queries based on the network/temporal pattern or the query content.

To further reinforce content indistinguishability, we randomly source the challenges from various question-answering platforms (e.g., Reddit, Quora, StackOverflow) and real user prompt datasets (e.g., WildChat [78]). To prevent proxies from simply checking if the seeds can be found from such sources, we prompt a local LLM like LLama-3.1-8B-Instruct[2] to stochastically rewrite the seeds with different wordings, topics, or languages. We empirically verify our method via an experiment: We simulate a proxy's knowledge with a small dataset of 800 WildChat prompts, 100 unknown audit prompts, and 100 known audit prompts. We then fine-tune a DeBERTaV3 model [26] via cross-validation on this dataset to predict whether a prompt is an audit and test on another dataset with the same composition but different content. The model only manages to achieve 0.5-0.6 test AUC, which is not better than random guessing. An alternative design is to drop end-to-end encryption for user queries, which would allow the coordinator to bypass the generation process above and audit actual user queries, but this would require users to put even more trust in the coordinator.

For verifying TLSNotary proofs, the coordinator hosts a local NodeJS server [56] that runs the TLSNotary Javascript API. As the API is written for the browser environment, to enable it in NodeJS, we use a virtual DOM and "monkey-patch" the browser-specific Web Worker API with a cross-platform alternative to load local files. After establishing the validity of the proof, the coordinator compares the original query and response with the selectively revealed text contained in the proof and checks the structure of the JSON response to ensure that there is no malicious manipulation.

[2]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

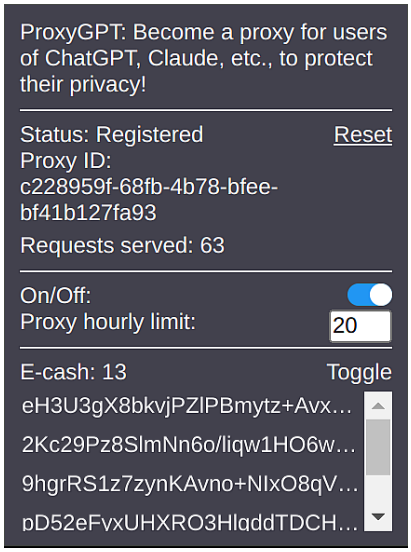

Fig. 5. Pop-up page for ProxyGPT's browser extension. It shows some basic statistics, allows users to easily turn the extension on/off and limit the number of proxy requests per hour, and displays the earned e-cash.

B. Proxies

1) Overview: We implement a browser extension for Chromium-based browsers using the ManifestV3 extension standard [24] to enable users to become proxies with minimal setup (Figure 5). The extension programmatically interacts with the chatbot websites using the provided GUI components in the same way as regular users do. Extension users must log in to the chatbot websites first and clear any bot detection mechanisms. To prevent Chrome from pausing unfocused browser tabs and ensure a smooth proxying process without hindering normal chatbot usage, extension users should dedicate a separate browser application window to each chatbot website without including additional tabs. Tor must also be installed and running in the background to enable communication between proxies and the coordinator (simply leaving the Tor browser open suffices).

2) Setup: Similar to the users, each proxy must generate an ECDH and ECDSA key pair, and then register with the coordinator using the generated keys. To store the private keys on the browser side, we set the Web Crypto API to generate them with the extractibility feature disabled (i.e., not exportable to a non-binary format even with Javascript), then store the entire binary private key object using the IndexedDB API [12]. The public keys, on the other hand, are exported to the SPKI format and base64-encoded.

Once registered, the proxy needs to authenticate itself by presenting a valid signature for a random nonce sent by the coordinator using the proxy's private ECDSA key. Upon successful validation of the proxy's signature, the coordinator issues to the proxy a signed JSON Web Token (JWT) [30] that confirms the proxy's authenticity. The proxy can subsequently present the JWT to the coordinator to retrieve new queries and submit responses until the token's expiration.

3) Handling queries: The proxy processes each query one at a time to avoid overloading the chatbots with several queries simultaneously. Each query is decrypted using the shared AES-GCM-256 key derived from the proxy's private ECDH key and the query owner's public ECDH key. For queries that are follow-ups of existing chat threads, the proxy must validate the ownership by verifying a signature produced by the query owners of the IDs of the latest queries in the threads. Note that the coordinator does not know whether a query is a new thread or part of an existing one since the payload is encrypted. The volume of queries served can also be controlled by the proxy.

In order for the proxy to be able to support multiple different chatbot websites while also maintaining consistency with real user experiences, we use a unified GUI-based query submission flow consisting of the following steps:

a. Create/Select chat thread: For a new query, locate the 'new chat' button/link. For a continuing query, locate the button/link for opening the chat thread with the relevant thread ID in the chat history. Click on the located element and wait ≈ 3 seconds for the website to load.
b. Find input area and enter query: The input area is either a textarea or a div with the attribute "contenteditable" set to true. After inputting the query into the input area, dispatch any relevant input event to make sure the internal state of the website is updated correctly, then wait 1 second for the submit button to appear.
c. Find submit button and click: Once the submit button appears and is usable, click on it and wait for 3 seconds.
d. Wait for response: Keep checking for DOM elements that indicate results are still streaming every 1 second.
e. Check for error: Chatbot providers may return server-side errors or enforce a rate limit
f. Retrieve query result: Once streaming is done, locate the DOM element corresponding to the latest message from the chatbot and extract the content (along with the generated thread ID in the updated URL).

Once the response has been retrieved, the proxy encrypts it with the derived AES key above and sends it to the coordinator. If any step above fails, the whole process is retried after a timeout of 1 minute.

4) Audit: Proxies handle verification challenges using TLSNotary's WebAssembly-based JavaScript API (which we further customized to enable easier redaction of sensitive information in JSON format). Proxies will participate in the three-way TLS protocol with our publicly hosted TLSNotary server, possibly over a VPN like Mullvad but not Tor due to the increased latency (once TLSNotary becomes sufficiently fast, Tor can be utilized to better protect proxy identity during audits). To connect to the chatbot server, proxies can either use our public WS proxy or host their own local WS proxy like noVNC's Websockify [51] to better preserve their privacy. Redacting sensitive information in the proof record involves hiding all HTTP request headers (the Request-URI in the Request-Line also needs to be partially hidden to avoid leaking the conversation ID) as well as parsing the JSON response to reveal only the relevant JSON structure and the content of the chatbot conversation (Figure 8). During this audit stage, proxies will not be able to serve any user requests.

5) E-cash: We introduce an incentive mechanism where proxies can obtain a ProxyGPT e-cash from the coordinator by successfully completing integrity audits. We implement an e-cash library based on Chaum's RSA-based blind signature

scheme [6, 7] using NodeJS's native 'crypto' library with guidance from RFC 9474 [14]. To obtain e-cash, proxies first generate an appropriate random message, *blind* it using the coordinator's e-cash public key, then attach the blinded result to the audit responses. If the audit passes, the coordinator signs the blinded message using their private e-cash key and sends the signature back to the proxy. The proxies can now *unblind* the coordinator's signature to obtain a different signature that is valid but unknown to the coordinator. The resulting signature and the generated random message together form the e-cash (Figure 5). This scheme thus allows ProxyGPT e-cash to be spent without revealing the identities of the original owners. When a coin is presented as payment, the coordinator must validate and verify that the coin has not been used.

6) Proxy Statistics: We present the following (hourly aggregated) statistics to users to help them choose proxies:

- Service Level Agreement (SLA) compliance rate: We define the SLA as a proxy finishing a user request in less than 1 minute. The rate of SLA compliance can let users know how likely they will get a response within 1 minute.
- Mean time to respond (MTTR): MTTR is defined as the time it takes for a proxy to respond, averaged over all finished queries. Unprocessed queries are not included.
- Load: We calculate the average volume of requests sent to a proxy per day as well as the volume in the last hour. This shows how active or busy a proxy is.
- Downvote rate: Users can report or *downvote* bad responses to help other users avoid low-quality proxies. This simple mechanism complements our more expensive cryptographic proxy verification technique.

VI. Evaluation

In this section, we evaluate our ProxyGPT proof-of-concept, focusing on its latency. The coordinator is implemented as a Tor hidden service with a user-friendly website (Figure 4), the notary as a TLSNotary server, and the proxy as a Chrome extension (Figure 5).

A. Query latency

We send 30 different queries[3] via our ProxyGPT website to a ChatGPT proxy running ChatGPT 3.5 in the Google Chrome browser on an Ubuntu 20.04 machine with 16GB of memory. We measure the time taken for the following:

- User delivering query to proxy: Involves the user submitting a request to the coordinator, who verifies the chosen proxy's identity and sends the query to the proxy.
- Proxy interacting with ChatGPT: Involves the proxy interacting with ChatGPT's website and notifying the extension's backend of the query results.
- Proxy delivering response to user: Involves the proxy sending the result to the coordinator, who verifies the proxy's identity and sends the response to the user.

Each query has the format "Tell me about <subject> in (one | two | three) paragraph(s)", where the subjects are chosen from a list of countries in the world, and the number of paragraphs is to induce responses of varying lengths (roughly between ASCII 800-1600 characters). Overall, it takes 15.41 ± 1.34 seconds on average for the user to receive the response for a query, with roughly half of the delay (8.3 ± 0.46 seconds) coming from the proxy's interaction with ChatGPT UI and the remaining half (≈ 7.1 seconds) coming from the Tor network communication between users, coordinator, and proxies (Table II). Of the 8 seconds spent interacting with ChatGPT's UI, about 2-3 seconds are idling time that we set to ensure that all UI operations and side-effects are fully propagated throughout ChatGPT website's internals correctly, while the remaining 5 seconds are the time for ChatGPT to return a complete response. Thus, using ProxyGPT incurs an additional 200% latency overhead compared to using ChatGPT normally. The longer the length of ChatGPT's response, the higher the latency (Figure 6), as the proxy has to spend more time waiting for the full response. From these results, we estimate that a single proxy can serve 6-8 queries per minute, and a single user can ask 3-4 queries per minute.

[3]This number is sufficient given our small margins of error calculated with a 95% t-distribution confidence interval (Tables II and III)

TABLE II. Query latency when using ProxyGPT with ChatGPT (sample size = 30).

Activity	Avg. time (seconds)	Standard deviation	Margin of error	Percent. of total
User delivering query to proxy	3.77	1.79	0.67	24.47%
Proxy interacting with ChatGPT	8.30	1.24	0.46	53.86%
Proxy delivering response to proxy	3.34	1.98	0.74	21.67%
Total wait time for user	15.41	3.58	1.34	100%
Estimated overhead vs. regular ChatGPT	≈ 10	N/A	N/A	$\approx 200\%$

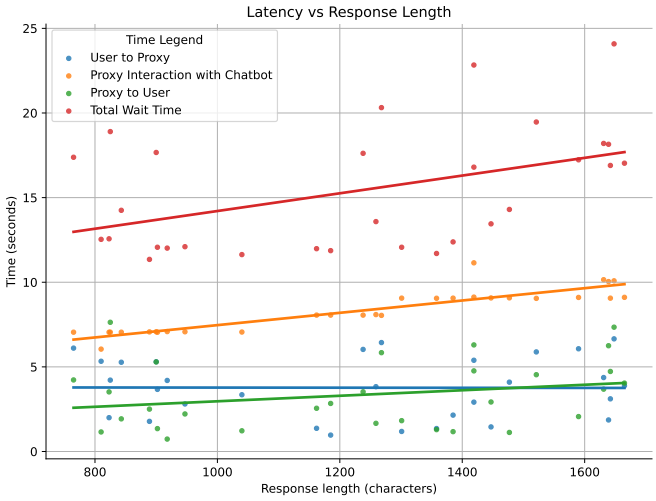

Fig. 6. Breakdown of query latency vs ChatGPT's response length (in characters). Pearson's correlation coefficient between total wait time and response length is 0.4487, indicating a moderate positive correlation.

TABLE III. AUDIT LATENCY (IN SECONDS) WITH TLSNOTARY AND CHATGPT DATA (SAMPLE SIZE = 10 FOR EACH LOCATION).

Distance from notary (km)	Mean	Median	Max	Standard deviation	Margin of error
≤ 1 (no VPN)	101.8	103	104	2.8	2.0
120	107	106	113	3.4	2.4
200	111.6	112	116	2.7	1.9
555	129.2	130	131	2.2	1.6
800	130.2	130	131	0.8	0.6
>800	Timeout	N/A	N/A	N/A	N/A

B. Audit latency

We also run a small-scale measurement of the total amount of time for a proxy to finish integrity audits with TLSNotary and ChatGPT. Note that proxies have to use a VPN instead of Tor to participate in the TLSNotary protocol due to its high overhead. To simulate the physical distance between a proxy and the notary server, we use Mullvad VPN [45] and run 10 audits for each chosen VPN server. On average, we find that the amount of time it takes to execute a single TLSNotary audit is typically between 100 and 130 seconds, with a moderate amount of variability depending on the proxy's network conditions and available hardware (Table III). Longer distances between the proxy and the notary lead to longer latency and can cause the proxy to be unable to send its MPC-encrypted message to the chatbot server due to timeout. In an actual production deployment, the notary can be an independent service with multiple servers deployed globally to increase coverage. More speed optimizations for TLSNotary are also under active development as it is still a nascent technology.

VII. SECURITY ANALYSIS & ETHICAL CONSIDERATIONS

Here, we examine potential vulnerabilities concerning the privacy, integrity, and other practical security aspects of ProxyGPT, along with ethical issues arising from its use.

A. Privacy

1) User identity: ProxyGPT protects user identity at the network level by separating the query from the source and relying on voluntary or paid proxies to interact with chatbots on the query author's behalf. This, however, does not guarantee absolute anonymity for users. Since we rely extensively on AC systems, the privacy protection offered is only as good as the strength of the employed AC protocols. Tor in particular is known to be vulnerable to adversaries capable of global traffic analysis [34]. Swapping out Tor for stronger AC systems is possible but will come at the expense of latency, which can be a deal-breaker for many chatbot users.

Another vector for determining user identity is via the content of the queries submitted. As discussed in Section III-A, chatbot providers can hypothetically analyze their database of known users to build a profile for each person, then try to match queries to their author via stylometry or machine learning techniques [46, 59]. More craftily, they can inject special watermarks into their chatbot responses and later try to find these watermarks in any known users' queries to link the anonymous queries and the authors together [35]. This can occur if users accidentally copy-paste the watermarked responses into their regular non-private chatbot conversations. To prevent this attack vector, users could locally perform query rewriting and sanitization [10] before requesting service from ProxyGPT. However, existing text anonymization approaches [61] such as sensitive/identifying term redaction and generalization do not consider the utility of the queries and are not efficient enough for interactive conversations. Whether it is possible to anonymize chatbot queries effectively remains an open challenge that we leave for future work.

2) Proxy identity: While ProxyGPT users can enjoy the network-level identity separation, proxies are still required to personally interact with chatbots using their own accounts or API keys. Consequently, hiding the identity of proxies is practically impossible if proxies have to deliver users' queries without any modification. As mentioned in III-C, chatbot providers can either collude with users or disguise themselves as users to ask uniquely identifiable queries. One possible defense that proxies can employ is to obfuscate the query content to reduce the chance of success for such linkage attack, but this has no formal guarantees. Even if the query content can be perfectly obfuscated, another vector of attack for chatbot providers is to passively analyze account activities or actively probe our system (e.g., sending ProxyGPT queries to proxies at specific times). Chatbot websites can also try to detect the presence of our extension, e.g., by checking for web-accessible resources or modifications from the extension [33].

B. Integrity

1) Proxy integrity: The use of TLSNotary to perform regular integrity audits can prevent proxies from misbehaving to an extent but does not completely rule out such foul play. A malicious proxy could try to distinguish audit queries from real user queries and only act honestly when handling the former, but this can be practically prevented with careful query generation (Section V-A3). Assuming audit queries are indistinguishable from regular ones, we can model this scenario as a 2-player non-cooperative game [47] where given a query, the proxy can choose to respond honestly with probability p_h and the coordinator can choose to audit with probability p_a (see Appendix A for full analysis).

2) Chatbot integrity: Generally, it is entirely within chatbot providers' rights to modify how their chatbot service behaves. They might intentionally provide subpar service to users suspected of participating in ProxyGPT as a punitive measure. Detecting such behavior can be challenging and would require collaboration among proxies to share and analyze their chatbot usage statistics together, which also needs to be done in a privacy-preserving manner. However, this risk should be low given the potential impact on the services' reputation if such a practice was discovered.

C. Security

1) Key management: Securely storing sensitive information on the browser side is generally difficult to achieve. Our browser extension relies on current native browser technologies, particularly the Web Crypto API and the IndexedDB API, to generate and store private keys locally on the browser side [12]. While this approach prevents the private keys from being exported to more accessible formats in JavaScript, it does

not prevent malicious actors with access to the proxy's browser from using the key or extracting information from the raw binary using other tools. If a cross-site scripting (XSS) attack is successfully executed on the extension, then the private keys can be stolen and abused. To reduce the risk of XSS, our implementation sanitizes any inputs coming from users and proxies before displaying them with DOMPurify [13].

2) DoS and Sybil attacks: In addition to the built-in DoS prevention mechanisms in the Tor network [72], our give-and-take e-cash economy can also be considered a form of DoS defense thanks to its reliance on completing TLSNotary-based audits, which have high computational costs. Sybil attacks, however, are harder to prevent due to the anonymous nature of ProxyGPT. If a dedicated actor floods our system with a large number of proxies, they can collect a large number of e-cash over an extended period of time, which can then be used to ask many queries. Consequently, they can collect a sizeable volume of user queries as well as chatbot responses and also affect the reputation and load of other proxies. The use of e-cash and TLSNotary can prolong the time it takes for an adversary to achieve such an influence on our system.

D. Ethical Considerations

1) Automation: Currently, most chatbot providers do not permit automated usage of their chatbot services except for API usage. As such, our browser extension's site manipulation could be considered a violation. However, these providers also delegate all rights to the content created on their platform to the users, from whom we explicitly obtain permission to extract their data. Furthermore, our implementation is a proof-of-concept for research purposes only, not for production or commercial use. Our give-and-take economy model via our e-cash system also minimizes the burden on the chatbot services. A full discussion of the legality of automated data extraction is beyond our paper's scope, but we believe that achieving privacy is as "justifiable" as the chatbot providers' scraping (copyrighted) web data for training LLMs [25].

2) Misuse: As with any anonymity services (or any technology in general), ProxyGPT can be exploited by malicious actors to execute inappropriate or even illegal activities. For instance, bad users can ask the chatbots how to create malware or organize a phishing campaign without having to create an identity-verified account [53]. Such queries may accidentally implicate the proxies, especially since proxy identities can be uncovered by the chatbot providers. We believe, however, that chatbot providers should be primarily responsible for aligning their chatbots with human interests and preventing them from responding to bad queries in the first place [57]. While it is still possible to "jailbreak" chatbots and bypass such safety mechanisms (e.g., via adversarial modifications to the query content [74]), aligning chatbots directly tackles the root cause of misuse and is also more achievable than policing human online activities.

3) Beyond chatbots: While the design of ProxyGPT is focused on chatbot services, our approach is broadly applicable to any web applications that require user identities. For example, on the LinkedIn platform [42], if user A views user B's profile, B will know that A has done so unless A changes to a stricter privacy setting that limits some of A's available features. User A can instead use a proxy-based system to view B's profile without revealing themselves. The use of such technology, however, must be considered carefully on a case-by-case basis to avoid harming the privacy of other users.

VIII. Conclusion and Future Work

In this paper, we present ProxyGPT–a novel proxy-based privacy-enhancing system for AI chatbot users that leverages the power of volunteers to separate a user's identity from their queries. Our system utilizes TLSNotary for proxy integrity and Chaum's e-cash for sustainability. Below, we detail possible next steps for our system. We hope that our work will inspire chatbot providers to build their services with privacy at the forefront, especially as this powerful technology becomes more intertwined with everyday lives.

Latency improvements: ProxyGPT's current query latency is dominated by two major components: the AC protocol's and the chatbot service's latency. While our implementation relies on Tor, which can be rather slow, our design is not restricted to Tor. Faster alternatives such as those based on the recent MASQUE protocol [15, 29] can potentially be used to further reduce the AC's delay (once the technology is ready). We can further reduce the *perceived* delay by transmitting parts of the chatbot response instead of waiting for it to be completely generated, similar to how current chatbots stream the responses token by token, but in larger chunks. Query latency is also affected by integrity audits which occur randomly during proxying. Reduction in audit latency depends on improvements in the underlying data provenance protocol, which is beyond the scope of our paper. There is much active research focusing on optimizing data provenance speed [5, 18, 39], but few public-ready implementations.

Trust assumptions: ProxyGPT currently assumes that the coordinator is a TTP and relies heavily on the coordinator for proxy management and user-proxy communication. While this centralized approach has several advantages such as easier implementation and proxy integrity checks, the coordinator also becomes a single point of failure for both system integrity and availability. Due to the difficult-to-verify nature of chatbot response data, fully decentralizing the functionalities of the coordinator while efficiently ensuring proxy integrity can be challenging. That said, we believe a truly decentralized ProxyGPT is worth investigating as it has the potential to promote user trust further. It could be argued that with our system, users would simply be moving their trust from the service providers to the proxies and the coordinator. However, we note an important distinction: ProxyGPT users only entrust our system with their chat conversation content, not their identities. Without ProxyGPT's identity separation, users would *always* be identifiable by chatbot providers, thus forcing them to trust that this information will never be exploited.

Additional features: We intend to incorporate more chatbots such as Gemini and Meta AI, as well as enable multi-modality with files, images, and audio chat. We also plan to include an in-browser LLM-powered content analysis module to help users proactively identify and prevent sensitive data leakage in their chatbot conversations [10, 68, 70] and to assist proxies with filtering out problematic queries [20].

REFERENCES

[1] AnonChatGPT, "Anonchatgpt," https://anonchatgpt.com/, n.d.

[2] Anthropic, "Introducing claude," https://www.anthropic.com/news/introducing-claude, 2023.

[3] C. Bocovich, A. Breault, D. Fifield, Serene, and X. Wang, "Snowflake, a censorship circumvention system using temporary WebRTC proxies," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 2635–2652. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/bocovich

[4] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying memorization across neural language models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=TatRHT_1cK

[5] S. Celi, A. Davidson, H. Haddadi, G. Pestana, and J. Rowell, "DiStefano: Decentralized infrastructure for sharing trusted encrypted facts and nothing more," Cryptology ePrint Archive, Paper 2023/1063, 2023. [Online]. Available: https://eprint.iacr.org/2023/1063

[6] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Boston, MA: Springer US, 1983, pp. 199–203. [Online]. Available: https://doi.org/10.1007/978-1-4757-0602-4_18

[7] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Advances in Cryptology — CRYPTO' 88*, S. Goldwasser, Ed. New York, NY: Springer New York, 1990, pp. 319–327. [Online]. Available: https://doi.org/10.1007/0-387-34799-2_25

[8] T. Chen, H. Bao, S. Huang, L. Dong, B. Jiao, D. Jiang, H. Zhou, J. Li, and F. Wei, "THE-X: Privacy-preserving transformer inference with homomorphic encryption," in *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3510–3520. [Online]. Available: https://aclanthology.org/2022.findings-acl.277

[9] B. Chesneau, "Gunicorn," https://gunicorn.org/, n.d.

[10] C. J. Chong, C. Hou, Z. Yao, and S. M. S. Talebi, "Casper: Prompt sanitization for protecting user privacy in web-based large language models," 2024. [Online]. Available: https://arxiv.org/abs/2408.07004

[11] J. Clusmann, F. R. Kolbinger, H. S. Muti, Z. I. Carrero, J.-N. Eckardt, N. G. Laleh, C. M. L. Löffler, S.-C. Schwarzkopf, M. Unger, G. P. Veldhuizen, S. J. Wagner, and J. N. Kather, "The future landscape of large language models in medicine," *Communications Medicine*, vol. 3, no. 1, pp. 1–8, Oct. 2023, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s43856-023-00370-1

[12] F. Corella and K. Lewison, "Storing cryptographic keys in persistent browser storage," Presented in *International Cryptographic Module Conference 2017 (ICMC '17)*, 2017, available at https://pomcor.com/documents/KeysInBrowser.pdf.

[13] cure53, "Dompurify," https://github.com/cure53/DOMPurify, n.d.

[14] F. Denis, F. Jacobs, and C. A. Wood, "RSA Blind Signatures," RFC 9474, Oct. 2023. [Online]. Available: https://doi.org/10.17487/RFC9474

[15] P. Dikshit, J. Sengupta, and V. Bajpai, "Recent trends on privacy-preserving technologies under standardization at the ietf," *SIGCOMM Comput. Commun. Rev.*, vol. 53, no. 2, p. 22–30, jul 2023. [Online]. Available: https://doi.org/10.1145/3610381.3610385

[16] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. USA: USENIX Association, 2004, p. 21. [Online]. Available: https://doi.org/10.5555/1251375.1251396

[17] J. Domingo-Ferrer, M. Bras-Amorós, Q. Wu, and J. Manjón, "User-private information retrieval based on a peer-to-peer community," *Data & Knowledge Engineering*, vol. 68, no. 11, pp. 1237–1252, 2009. [Online]. Available: https://doi.org/10.1016/j.datak.2009.06.004

[18] J. Ernstberger, J. Lauinger, Y. Wu, A. Gervais, and S. Steinhorst, "ORIGO: Proving provenance of sensitive data with constant communication," Cryptology ePrint Archive, Paper 2024/447, 2024. [Online]. Available: https://eprint.iacr.org/2024/447

[19] F5, "Nginx," https://www.nginx.com/, n.d.

[20] I. Fedorov, K. Plawiak, L. Wu, T. Elgamal, N. Suda, E. Smith, H. Zhan, J. Chi, Y. Hulovatyy, K. Patel, Z. Liu, C. Zhao, Y. Shi, T. Blankevoort, M. Pasupuleti, B. Soran, Z. D. Coudert, R. Alao, R. Krishnamoorthi, and V. Chandra, "Llama Guard 3-1B-INT4: Compact and Efficient Safeguard for Human-AI Conversations," 2024. [Online]. Available: https://arxiv.org/abs/2411.17713

[21] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Mayer, "Consistent, yet anonymous, web access with lpwa," *Communications of the ACM*, vol. 42, no. 2, pp. 42–47, 1999. [Online]. Available: https://doi.org/10.1145/293411.293447

[22] E. Gabber, P. B. Gibbons, Y. Matias, and A. Mayer, "How to make personalized web browsing simple, secure, and anonymous," in *Financial Cryptography: First International Conference, FC'97 Anguilla, British West Indies February 24–28, 1997 Proceedings 1*. Springer, 1997, pp. 17–31. [Online]. Available: https://doi.org/10.1007/3-540-63594-7_64

[23] Google, "Introducing gemini: Our largest and most capable ai model," https://blog.google/technology/ai/google-gemini-ai/, 2023.

[24] ——, "Manifest v3," https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3, n.d.

[25] M. M. Grynbaum and R. Mac, "New york times sues openai and microsoft over a.i. use of copyrighted work," https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html, 2023.

[26] P. He, J. Gao, and W. Chen, "DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=sE7-XhLxHA

[27] Hola VPN Ltd., "Hola," https://hola.org/, n.d.

[28] HuggingFace, "Huggingchat," https://huggingface.co/chat/, n.d.

[29] IETF, "Multiplexed application substrate over quic encryption," https://datatracker.ietf.org/wg/masque/about/, n.d.

[30] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7519

[31] R. Jones, R. Kumar, B. Pang, and A. Tomkins, "I know what you did last summer: Query logs and user privacy," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, ser. CIKM '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 909–914. [Online]. Available: https://doi.org/10.1145/1321440.1321573

[32] M. Kalka and M. Kirejczyk, "A comprehensive review of TLSNotary protocol," 2024. [Online]. Available: https://arxiv.org/abs/2409.17670

[33] S. Karami, P. Ilia, K. Solomos, and J. Polakis, "Carnus: Exploring the privacy threats of browser extension fingerprinting." in *In Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, USA: The Internet Society, 2020. [Online]. Available: https://doi.org/10.14722/ndss.2020.24383

[34] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. K. Jha, "De-anonymisation attacks on tor: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2324–2350, 2021. [Online]. Available: https://doi.org/10.1109/COMST.2021.3093615

[35] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. Honolulu, HI, USA: PMLR, 23–29 Jul 2023, pp. 17 061–17 084. [Online]. Available: https://proceedings.mlr.press/v202/kirchenbauer23a.html

[36] J. Lai, W. Gan, J. Wu, Z. Qi, and P. S. Yu, "Large language models in law: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2312.03718

[37] Y. Li, S. Wang, H. Ding, and H. Chen, "Large Language Models in Finance: A Survey," in *4th ACM International Conference on AI in Finance*. Brooklyn NY USA: ACM, Nov. 2023, pp. 374–382. [Online]. Available: https://dl.acm.org/doi/10.1145/3604237.3626869

[38] Z. Li, C. Liang, J. Peng, and M. Yin, "The Value, Benefits, and Concerns of Generative AI-Powered Assistance in Writing," in *Proceedings of the CHI Conference on Human Factors in Computing*

Systems, ser. CHI '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–25. [Online]. Available: https://doi.org/10.1145/3613904.3642625

[39] Z. Luo, Y. Jia, Y. Shen, and A. Kate, "Proxying is enough: Security of proxying in TLS oracles and AEAD context unforgeability," Cryptology ePrint Archive, Paper 2024/733, 2024, https://eprint.iacr.org/2024/733. [Online]. Available: https://eprint.iacr.org/2024/733

[40] Meta, "Meta ai," https://ai.meta.com/meta-ai/, n.d.

[41] Microsoft, "Introducing microsoft 365 copilot – your copilot for work," https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/, 2023.

[42] ——, "Linkedin," https://linkedin.com, n.d.

[43] N. Mireshghallah, M. Antoniak, Y. More, Y. Choi, and G. Farnadi, "Trust no bot: Discovering personal disclosures in human-LLM conversations in the wild," in *First Conference on Language Modeling*, 2024. [Online]. Available: https://openreview.net/forum?id=tIpWtMYkzU

[44] N. Mireshghallah, H. Kim, X. Zhou, Y. Tsvetkov, M. Sap, R. Shokri, and Y. Choi, "Can LLMs keep a secret? testing privacy implications of language models via contextual integrity theory," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=gmg7t8b4s0

[45] Mullvad, "Mullvad vpn," https://mullvad.net/, n.d.

[46] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song, "On the feasibility of internet-scale author identification," in *2012 IEEE Symposium on Security and Privacy*. Los Alamitos, CA, USA: IEEE Computer Society, may 2012, pp. 300–314. [Online]. Available: https://doi.org/10.1109/SP.2012.46

[47] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951, https://doi.org/10.2307/1969529.

[48] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," 2023. [Online]. Available: https://arxiv.org/abs/2311.17035

[49] M. Nasr, H. Zolfaghari, A. Houmansadr, and A. Ghafari, "Massbrowser: Unblocking the censored web for the masses, by the masses." in *27th Annual Network and Distributed System Security Symposium (NDSS '2020)*. San Diego, CA, USA: The Internet Society, 2020. [Online]. Available: https://doi.org/10.14722/ndss.2020.24340

[50] S. Neel and P. Chang, "Privacy issues in large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2312.06717

[51] noVNC, "Websockify," https://github.com/novnc/websockify, n.d.

[52] OpenAI, "Introducing chatgpt," https://openai.com/blog/chatgpt, 2022.

[53] ——, "Disrupting malicious uses of ai by state-affiliated threat actors," https://openai.com/blog/disrupting-malicious-uses-of-ai-by-state-affiliated-threat-actors, 2024.

[54] ——, "Memory and new controls for chatgpt," https://openai.com/index/memory-and-new-controls-for-chatgpt/, 2024.

[55] ——, "Start using chatgpt instantly," https://openai.com/index/start-using-chatgpt-instantly/, 2024.

[56] OpenJS, "Nodejs," https://nodejs.org/, n.d.

[57] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 27 730–27 744. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

[58] Pallets, "Flask," https://flask.palletsprojects.com/en/3.0.x/, n.d.

[59] S. T. Peddinti and N. Saxena, "Web search query privacy: Evaluating query obfuscation and anonymizing networks," *J. Comput. Secur.*, vol. 22, no. 1, p. 155–199, jan 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2590636.2590640

[60] Perplexity, "Perplexity.ai," https://www.perplexity.ai/, n.d.

[61] I. Pilán, P. Lison, L. Øvrelid, A. Papadopoulou, D. Sánchez, and M. Batet, "The Text Anonymization Benchmark (TAB): A Dedicated Corpus and Evaluation Framework for Text Anonymization," *Computational Linguistics*, vol. 48, no. 4, pp. 1053–1101, 12 2022. [Online]. Available: https://doi.org/10.1162/coli_a_00458

[62] Privacy and Scaling Explorations, "Tlsnotary," https://tlsnotary.org/, n.d.

[63] Private-AI, "Privategpt," https://www.private-ai.com/private-chatgpt/, n.d.

[64] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[65] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, "Code llama: Open foundation models for code," 2024. [Online]. Available: https://arxiv.org/abs/2308.12950

[66] S. Sasy and I. Goldberg, "Sok: Metadata-protecting communication systems," Cryptology ePrint Archive, Paper 2023/313, 2023. [Online]. Available: https://eprint.iacr.org/2023/313

[67] F. Shirazi, M. Simeonovski, M. R. Asghar, M. Backes, and C. Diaz, "A survey on routing in anonymous communication protocols," *ACM Comput. Surv.*, vol. 51, no. 3, jun 2018. [Online]. Available: https://doi.org/10.1145/3182658

[68] L. Siyan, V. C. Raghuram, O. Khattab, J. Hirschberg, and Z. Yu, "Papillon: Privacy preservation from internet-based and local language model ensembles," 2024. [Online]. Available: https://arxiv.org/abs/2410.17127

[69] R. Staab, M. Vero, M. Balunovic, and M. Vechev, "Beyond memorization: Violating privacy via inference with large language models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=kmn0BhQk7p

[70] R. Staab, M. Vero, M. Balunović, and M. Vechev, "Large language models are advanced anonymizers," 2024. [Online]. Available: https://arxiv.org/abs/2402.13846

[71] TeroBox, "Terobox," https://terobox.com, n.d.

[72] Tor, "Onion service dos guidelines," https://community.torproject.org/onion-services/advanced/dos/, 2021.

[73] W3C, "Web cryptography api," https://w3c.github.io/webcrypto/, 2023.

[74] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 80 079–80 110. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/fd6613131889a4b656206c50a8bd7790-Paper-Conference.pdf

[75] xAI, "Announcing grok," https://x.ai/blog/grok, 2023.

[76] You.com, "Introducing youchat," https://about.you.com/introducing-youchat-the-ai-search-assistant-that-lives-in-your-search-engine-eff7badcd655/, 2023.

[77] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1919–1938. [Online]. Available: https://doi.org/10.1145/3372297.3417239

[78] W. Zhao, X. Ren, J. Hessel, C. Cardie, Y. Choi, and Y. Deng, "Wildchat: 1m chatGPT interaction logs in the wild," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=Bl8u7ZRlbM

[79] J. Zhou, E. Xu, Y. Wu, and T. Li, "Rescriber: Smaller-llm-powered user-led data minimization for navigating privacy trade-offs in llm-based conversational agent," 2025. [Online]. Available: https://arxiv.org/abs/2410.11876

APPENDIX

A. Proxy Integrity Analysis

Consider the following reward matrix for a malicious proxy and a coordinator:

TABLE IV. GENERIC REWARD MATRIX. THE NUMBERS ON THE LEFT OF THE TUPLES ARE THE REWARDS FOR THE COORDINATOR, AND THE ONES ON THE RIGHT ARE FOR THE PROXY.

Coord. \ Proxy	Honest	Dishonest
Audit	(r^c_{ah}, r^p_{ah})	(r^c_{ad}, r^p_{ad})
Non-audit	(r^c_{nh}, r^p_{nh})	(r^c_{nd}, r^p_{nd})

We can simplify the matrix by fixing the rewards of the worst and best possible outcomes for each player. More specifically, the malicious proxy wins if it acts dishonestly with a non-audit query ($r^p_{nd} = 1$) and loses if the query is an audit one ($r^p_{ad} = -1$). The coordinator wins if it catches the proxy being dishonest ($r^c_{ad} = 1$) and loses if it fails to ($r^c_{nd} = -1$). In addition, the coordinator does not get any reward if it does not perform an audit and the proxy is honest ($r^c_{nh} = 0$).

TABLE V. SIMPLIFIED REWARD MATRIX.

Coord. \ Proxy	Honest	Dishonest
Audit	(r^c_{ah}, r^p_{ah})	$(1, -1)$
Non-audit	$(0, r^p_{nh})$	$(-1, 1)$

We further assume the following inequalities:

- $-1 < r^c_{ah} \leq 0$: Performing audits can cost the coordinator, but is less costly than missing a dishonest query.
- $r^p_{ah} \geq 0$: Being honest with an audit query can reward the proxy even though the proxy has to spend extra computation resources to complete the audit.
- $-1 < r^p_{nh} \leq 0$: Being honest with a non-audit query can be costly since the proxy has to spend computation resources without being rewarded. However, it is less costly than being caught dishonest and banned from participating.

Let p_a be the probability of auditing, and let p_h be the probability of the proxy being honest. The state where no players can improve their utility by changing their strategy is called *Nash Equilibrium* (NE). A pure-strategy NE is when the actions are chosen deterministically, whereas a mixed-strategy NE is when the actions are chosen stochastically.

We can see that no pure-strategy NE exists with the simplified reward scheme. The proxy's expected reward is $r^p_{ah}p_a + r^p_{nh}(1-p_a)$ if it is honest and $-p_a + (1-p_a) = 1-2p_a$ if it is dishonest. Thus, the proxy will mix between the two strategies if:

$$r^p_{ah}p_a + r^p_{nh}(1-p_a) = 1-2p_a \iff p_a = \frac{1-r^p_{nh}}{r^p_{ah} - r^p_{nh} + 2}$$

The coordinator's expected reward is $r^c_{ah}p_h + (1-p_h)$ if it audits and $p_h - 1$ if it does not. Thus, the coordinator will mix between the two strategies if:

$$r^c_{ah}p_h + (1-p_h) = p_h - 1 \iff p_h = 2/(2 - r^c_{ah})$$

Therefore, the mixed-strategy NE is $p^*_a = 1/(r^p_{ah} - r^p_{nh} + 2)$ and $p^*_h = 2/(2 - r^c_{ah})$. Since we assume that $-1 < r^c_{ah} \leq 0$, we have $2/3 < p^*_h \leq 1$.

The mixed-strategy NE expected reward for the proxy is:

$$E[R_P] = p^*_a(p^*_h r^p_{ah} - (1-p^*_h)) + (1-p^*_a)(p^*_h r^p_{nh} + (1-p^*_h))$$

The mixed-strategy NE expected reward for the coordinator is:

$$E[R_C] = p^*_a(p^*_h r^c_{ah} + (1-p^*_h)) - (1-p^*_a)(1-p^*_h) = \frac{r^c_{ah}}{2 - r^c_{ah}}$$

Since $-1 < r^c_{ah} \leq 0$, we have $-1/3 < E[R_C] \leq 0$.

Depending on the exact rewards, we have the following results for the mixed-strategy NE and expected rewards:

TABLE VI. MIXED-STRATEGY NE AND EXPECTED PAYOFF FOR DIFFERENT REWARD SCHEMES

Scenario	p^*_a	p^*_h	$E[R_P]$	$E[R_C]$
$r^p_{ah} = r^p_{nh} = r^c_{ah} = 0$	$1/2$	1	0	0
$r^p_{ah} = r^p_{nh} = 0, r^c_{ah} < 0$	$1/2$	$\frac{2}{2 - r^c_{ah}}$	0	$\frac{r^c_{ah}}{2 - r^c_{ah}}$
$r^p_{ah} > 0, r^p_{nh} = 0$	$\frac{1}{r^p_{ah} + 2}$	$\frac{2}{2 - r^c_{ah}}$	$\frac{r^p_{ah}}{r^p_{ah} + 2}$	$\frac{r^c_{ah}}{2 - r^c_{ah}}$
$r^p_{ah} = 0, r^p_{nh} < 0$	$\frac{1 - r^p_{nh}}{2 - r^p_{nh}}$	$\frac{2}{2 - r^c_{ah}}$	$\frac{r^p_{nh}}{2 - r^p_{nh}}$	$\frac{r^c_{ah}}{2 - r^c_{ah}}$
$r^p_{ah} > 0, r^p_{nh} < 0$	$\frac{1 - r^p_{nh}}{r^p_{ah} - r^p_{nh} + 2}$	$\frac{2}{2 - r^c_{ah}}$	$\frac{r^p_{ah} + r^p_{nh}}{r^p_{ah} - r^p_{nh} + 2}$	$\frac{r^c_{ah}}{2 - r^c_{ah}}$

Based on this, we can draw the following high-level guidelines when designing and implementing ProxyGPT:

- To reduce the need for audits (i.e., decrease p^*_a), we can increase the reward for proxies for successfully completing audits r^p_{ah}, such as by increasing the number of e-cash. Decreasing the cost of honest proxying r^p_{nh} is less straightforward since this is often a fixed constant (e.g., each proxied request costs the proxy one chatbot request). With careful design, we could focus on the unspent request bandwidth that would have been wasted.
- To encourage proxies to be honest (i.e., increase p^*_h), we can (try to) decrease r^c_{ah}, the cost for the coordinator to perform audits when the proxy is honest. This is largely dependent on the underlying TLS-backed data provenance protocol.

We note that the current analysis model only allows us to make relative interpretations concerning the behavior of the proxies and the coordinator. It does not enable us to determine the exact amount of the rewards needed to achieve a beneficial scenario. We would need a more detailed model that can, for example, correctly relate the two quantities r^p_{ah} and r^c_{ah} since giving more rewards to a proxy can introduce future costs to the system. We leave this for future work.

B. *Example of a Verification Challenge*

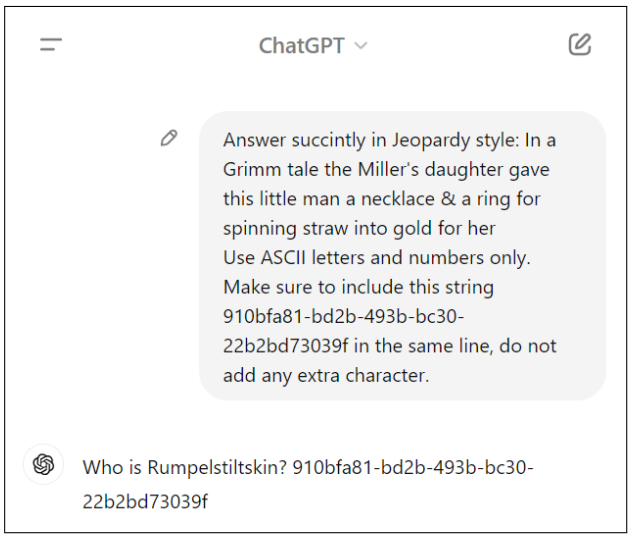

Fig. 7. Example of a verification challenge for proxies (screenshot taken from ChatGPT website).

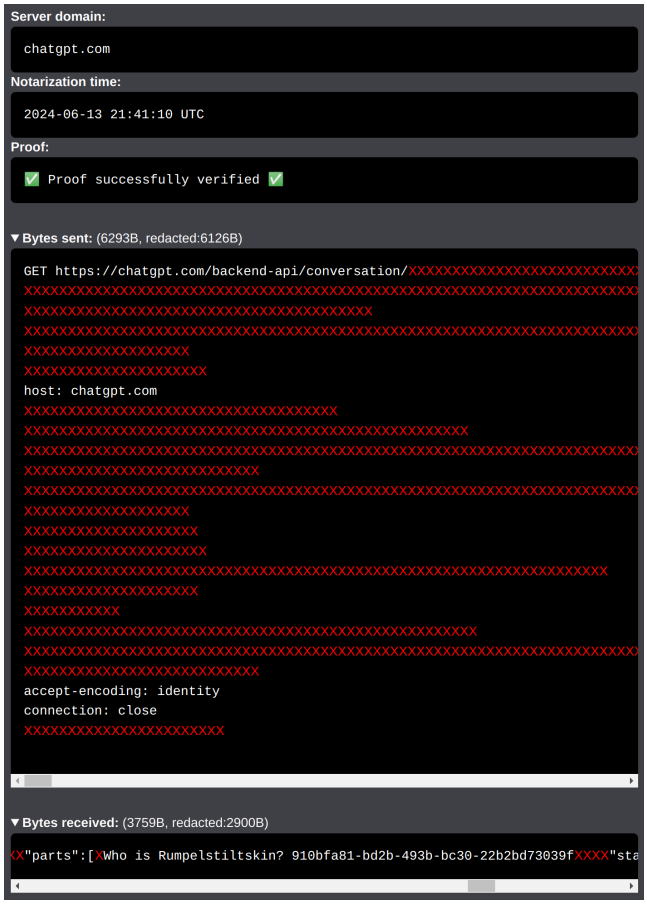

Fig. 8. Visualization of a verified TLSNotary proof (screenshot taken from https://tlsnotary.github.io/proof_viz/). All sensitive information such as the authorization token in the request header has been redacted by the proxy.