GotCHA Catch'Em All: A Decentralized CAPTCHA Approach

Mario Yaksetig
University of Porto
Porto, Portugal
mario.yaksetig@fe.up.pt

Duarte Fleming
University of Porto
Porto, Portugal
ee10076@fe.up.pt

Tiago Seabra
University of Porto
Porto, Portugal
ee12120@fe.up.pt

Abstract—**We propose a novel decentralized and high performance CAPTCHA architecture. Our approach relies on off-chain computation to allow nodes in charge of creating CAPTCHA challenges, which we call workers, to freely join and leave the platform at any point in time.**

Index Terms—**CAPTCHAs, blockchain, machine learning**

I. Introduction

In 1950, Alan Turing proposed the Turing Test [1]. This test, also referred to as the Imitation Game, is designed to determine whether or not machines can think, or appear to think, like humans. The Turing Test conjectures a scenario where an interrogator asks two participants, where one of the participants is a machine and the other is human, a series of questions. The interrogator does not know which one is which, and attempts to guess which participant is a machine. If the interrogator fails to determine which participant is human, then the machine passes the test and successfully impersonated the human.

Extending on Turing's research, John Searle introduced the Chinese room argument [2], which defines two variants of artificial intelligence (AIs): *weak AI* and *strong AI*. This Chinese room argument assumes that artificial intelligence research succeeded in creating a computer that behaves as if it has full understanding of the Chinese language and successfully passes the Turing test by convincing any Chinese speaker that the AI is indeed another human Chinese speaker. Given this setting, Searle attempts to answer if the machine indeed understands the Chinese language or, if the machine is simulating the ability to understand Chinese. If the machine understands the Chinese language, then it is a strong AI. If it is simulating the understanding, it is a weak AI. Searle then concludes that the strong AI option is not achievable and that an artificial intelligence must, by definition, be a weak AI.

Regardless of philosophy conjectures and controversial debates in the research focused on distinguishing AI from humans, the capabilities of AI agents evolved exponentially. This fast-paced growth, led to an enormous increase of robot (or bot) usage on the internet.

To mitigate the risk of bots acting as malicious actors, Luis von Ahn et. al [3] introduced the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). As implied by the name, a CAPTCHA is a practical and automated application of Turing's test, and allows a website owner to prevent bots from accessing their website or its contents. CAPTCHAs are based on the assumption that some underlying artificial intelligence problem is hard for computers, yet easy for humans (e.g., reading distorted letters). CAPTCHAs are presently widely deployed on the Internet to fight spam and protect against sybil attacks [4], where an entity creates a large number of pseudonym identities and attempts to subvert the security of a system.

CAPTCHA services, however, have a hidden terms of use clause that many are not aware of. CAPTCHA service providers treat humans as a part of a larger distributed system, where each person performs a small part of a larger *hidden* computation. For example, initial CAPTCHA challenges that required users to manually type the shown distorted letters, actually asked users to translate images of real words and numbers taken from archival texts. As a result, users contributed with a digitization service to an external party without awareness of such. As an alternative example, Google conducted an experiment [5] using its reCAPTCHA [6] spam-fighting system to improve the data in the Google Maps service by having users identify attributes such as street names and business addresses.

II. CAPTCHAs

CAPTCHAs are now ubiquitously found on the Internet to ensure that entities interacting with an online service are indeed human. While CAPTCHAs ensure that automated online access is reduced, "real" web users are traditionally forced to suffer through increasingly convoluted and unfriendly challenges that negatively impact their user experience and effectively steal labor hours without their knowledge.

CAPTCHA services, as most online services, have grown dependent of centralized points of control. Typically, a user accesses a web service and attempts to perform a specific action. The website, to verify whether or not the user is a 'bot', displays a CAPTCHA challenge that the user must successfully pass to prove its humanity by completing a human task. The user submits a response to the CAPTCHA service provider which then confirms whether the solution is correct; If considered valid, the website is informed that the entity attempting to access the service is a human and that it should be granted access. The hidden reality (see Fig 1) is that there exists an external entity benefiting from the work that a user provides to the CAPTCHA service provider. This collusion between the external party and CAPTCHA service provider results in a setting where the human work labor hours are effectively exploited.

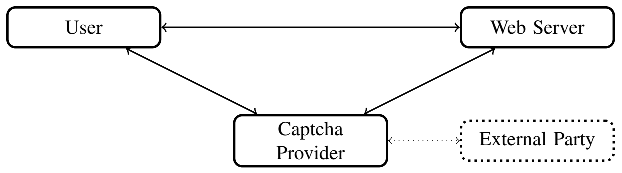

Fig. 1. Architecture Diagram of a traditional CAPTCHA service provider. We represent using the dotted lines the often hidden entities also involved in existing CAPTCHA interactions.

A. External CAPTCHA solving

Naturally, as CAPTCHAs evolved, the Internet adapted to attempt to bypass these challenges in more sophisticated manners. Presently, many mechanisms that rely on outsourcing the CAPTCHA solving process exist to bypass these challenges. Namely, a user attempting to outsource a CAPTCHA solving service can attempt to use an automated CAPTCHA solving service, a click farm, or even crowd-sourcing CAPTCHA solving methods.

1) Automated CAPTCHA Solving: Users attempting to outsource a CAPTCHA solving can use automated services powered by bots. For example, many of the letter-based CAPTCHA challenges are easily solved automatically using bots that can perform Optical Character Recognition (OCR). A big advantage of this approach, is that these OCR bots are typically able to solve a large number of these CAPTCHAs, thus resulting in a very cost-effective solution. Presently, however, that is no longer the case as the majority of the Internet CAPTCHA challenges have adapted to be more complex and cannot be easily subverted by bots with relatively simple optical vision capabilities.

2) Click farms: While there are a variety alternative ways to bypass CAPTCHA challenges (e.g., using specific computer algorithms, or trained machine learning models), human click farms [7] represent the most popular and effective solution. A CAPTCHA click farm is a business that employs people to solve these challenges as quickly as possible; This can be done manually or through the use of bots. While CAPTCHA click farms have been around for some time, they faced an increased scrutiny in recent years. Figure 2 shows an architecture overview of a CAPTCHA interaction that additionally involves a click farm service provider and human solver(s).

In practical terms, from the perspective of a malicious actor, the CAPTCHA problem does not represent much friction as they can cheaply outsource human labor to solve the provided challenges and bypass these security mechanisms as the tasks are actually performed by humans.

3) Crowd-sourcing CAPTCHA solving: In this setting, a web service $\mathcal{X}$ provides users with a CAPTCHA challenge upon registration on their platform. An adversary, controlling a different web service $\mathcal{Y}$, requests challenges from $\mathcal{X}$ and displays them as if they are valid challenges to use on the malicious web service $\mathcal{Y}$.

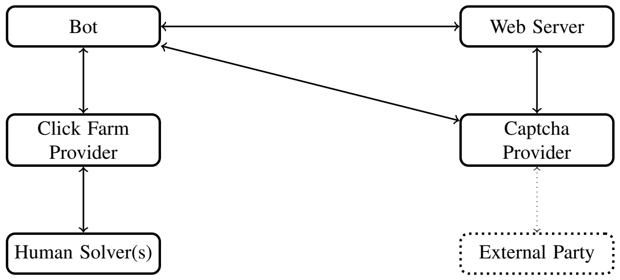

Fig. 2. Architecture overview of a traditional CAPTCHA interaction involving the outsourcing from a click farm.

This challenge, however, is requested by a bot attempting to register on a different website to perform potentially malicious actions. Examples such as [8], showcase a setting with spammers that created a game where users (unknowingly) solve CAPTCHA challenges on behalf of another party to make progress in the game.

a) Differences: The main difference between this problem, also known as the mafia-fraud, and the click farm is that in the latter, the CAPTCHA solvers are aware that they are solving a challenge on behalf of someone for a profit in return. In contrast, the solvers in the crowd-sourcing setting, believe that they are solving a CAPTCHA for a specific website when, in fact, this solving results in powering a bot to perform an action on a different website.

III. Problem Statement

Presently, CAPTCHA service providers that support the majority of the websites on the Internet are massively centralized and controlled by big tech companies. These companies are known for their privacy-invasive practices, and that is also true for the CAPTCHA challenges they provide. These existing approaches traditionally lack usability and typically frustrate many of the existing internet users.

Ironically, one can even find approaches—designed to compete against the big tech services—that boast its privacy awareness by design, yet collect information such as IP addresses, browser type, Internet service provider, platform type, device type, operating system, mouse movements, scroll position, key press events, touch events, and gyroscope / accelerometer information [9]. This data, which may even include cookies collected by third parties, is then used to determine human confidence scores.

In the current centralized CAPTCHA landscape, users interacting with CAPTCHA challenges are producing work that benefits directly the CAPTCHA provider, leaving a massive gap for a platform that is able to provide a more decentralized and privacy-oriented solution that focuses more on usability rather than on the training of machine learning models. This privacy-oriented and decentralized solution that focuses on usability is where our approach, called GotCHA , fits in.

GotCHA addresses the following challenge: "Without tracking any metrics that allow for the profiling or distinguishing of an individual, can we check if an entity trying to access a service is a person or a bot?". In this specific work, we focus on the architecture, rather than the types of challenges, that is required to ensure that the platform is able to scale to real-world needs while ensuring that there is no collecting of personally identifiable information. Additionally, we highlight that our proposal does not include or even consider solutions that rely on online credentials, search history inspection or external identification mechanisms.

We believe this is a more valuable contribution as it preserves the privacy of the parties accessing the services while allowing for a decentralized architecture where more parties can benefit directly from participating in the system.

IV. Protocol Overview

GotCHA is designed in a simple and modular manner. First, the user communicates with a web server, which contains the resources the user desires to access. The web server communicates with a smart contract that keeps a registry of the (decentralized) workers existing in the worker pool and that are able to provide a CAPTCHA challenge. The web server selects a specific worker to query, and requests a CAPTCHA challenge. The worker responds with the challenge, along with the different answer options, where one of the options is the right answer. The web server, upon receiving the CAPTCHA puzzle, displays—to the user—the challenge along with the different options. Finally, the user provides the solution to the puzzle, and the web server verifies the correctness of solution and acts accordingly.

V. System Architecture

Our design comprises five entities: a set of at least one user, a web server, a smart contract, a registration service, and a set of workers.

The users interact with the web server by providing requests for the content they desire to access.

The web server communicates with a smart contract that allocates individual workers to supply the web server with CAPTCHA challenges to show to the users, thus requiring users to prove that they are not automated and are indeed a human person.

The smart contract is in charge of adding and removing workers from the worker pool as well as selecting which workers are assigned to provide fresh challenges for the web servers. In practical terms, the smart contract acts as an intermediary between the web server(s) and the worker pool.

The registration service provides web servers with authentication tokens, or keys, that can later be used to request CAPTCHA challenges from the workers.

The worker pool is a decentralized pool of servers that are constantly creating new CAPTCHA challenges. We note that this entity requires an anti-sybil method for admission as, otherwise, a malicious actor can spin up an almost unlimited amount of workers and provide malicious challenges to the requesting web servers.

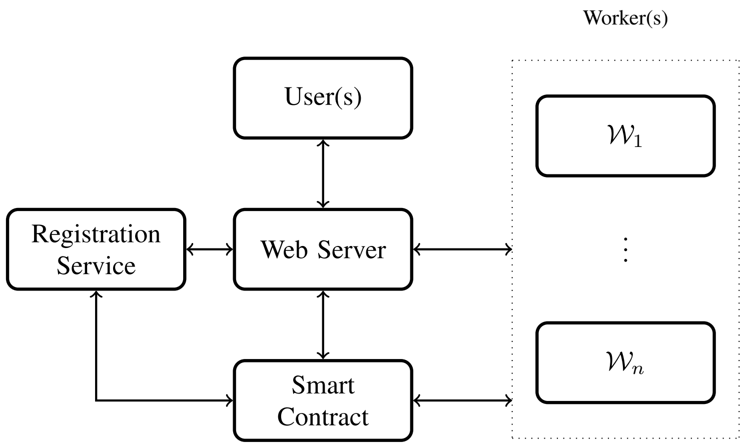

Fig. 3. GotCHA System Architecture.

VI. GotCHA

We now describe our decentralized CAPTCHA approach. For proper functioning of the design, we assume the existence of a pool of workers, where each worker creates a set of challenges and informs the smart contract that they are ready to provide challenges to requesting websites. A ladder diagram containing the protocol is exposed in Figure 4.

Step 1 (Registration): Before initialization of the system, web servers must register with the registration service, and workers must register with the smart contract. This step results in web servers having a valid authentication token to show workers that the requests are valid and ensures that web servers can have access to a public list containing all the workers that are registered in the system. Upon successful registration of a worker with the smart contract, a worker is considered active and ready for challenge provisioning.

Step 2 (Web Page Request): The user accesses the desired website, which is protected by a GotCHA challenge to detect whether or not the user is an automated entity or a real person.

Step 3 (Worker Lookup): The website, upon receiving an access request, proceeds to query the smart contract to request the information of the next assigned worker that is expected to provide fresh GotCHA challenges.

Step 4 (Worker Assignment): The smart contract, maintains a global state of the worker pool, and acts as a public-key infrastructure (PKI) for the web servers in the system. The web servers, keep a local copy of the worker pool and, given a specific time, perform a verifiable shuffle and selects a worker from the pool. Alternatively, the contract itself could perform the verifiable shuffle to select the next worker to provide the challenge to the web server. This approach, however, translates

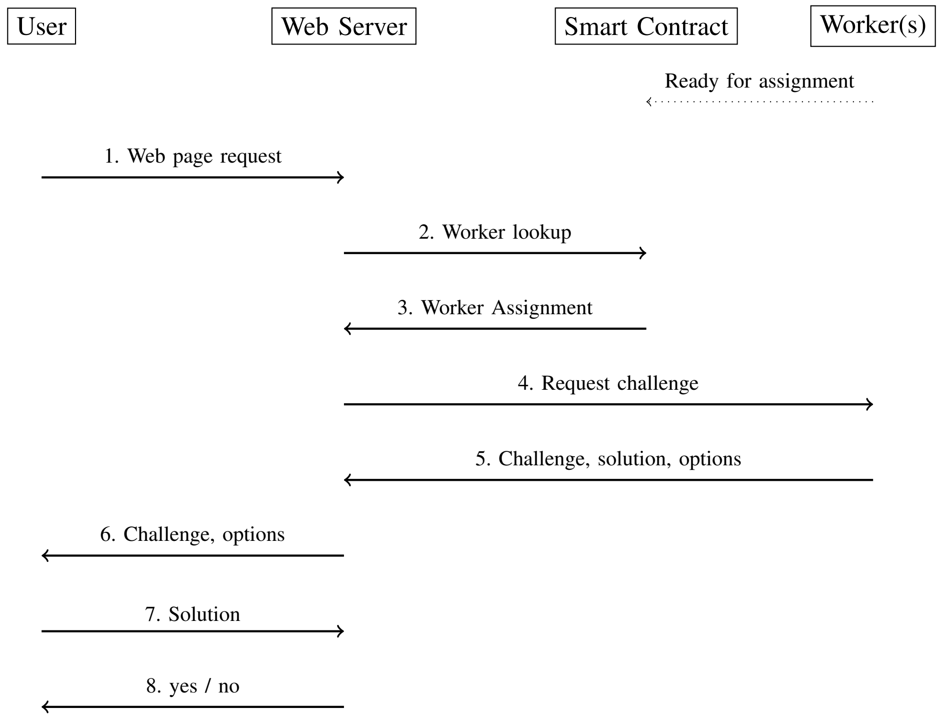

Fig. 4. GotCHA Protocol Overview.

into a bottleneck in a real world deployment that requires low latency and cheap execution costs.

We note that, in this step, the smart contract may keep additional metrics and perform data analysis to determine what the best worker for a specific request is. For example, it is probably better to assign a more reliable worker that is in a geographic region closer to the web server requesting a new challenge.

Step 5 (Challenge Request): The web server sends to the worker a request for a new GotCHA challenge to provide to the website access in an attempt to discover whether or not such access originates from a legitimate user or an automated script.

Step 6 (Challenge Provision): The worker responds to the web server request with a fresh GotCHA challenge along with the answer options. For example, a worker may provide a freshly generated cartoon representation of a duck, and four possible options (e.g., chicken, flamingo, duck, pigeon).

Step 7 (Challenge Display): The web server displays the GotCHA challenge to the end user along with the associated options for the user to select the appropriate answer.

Step 8 (Solution Submission): The user, upon visualizing the GotCHA challenge, solves the challenge, and provides the adequate response back to the web server for evaluation.

Step 9 (Solution Verification): The web server receives the solution submission from the user, and locally verifies whether or not the response to the GotCHA challenge is correct.

VII. Implementation

In this section, we describe our prototype implementation of the protocol, where we simulate a test network token faucet with a simple form for collecting the wallet address of a recipient expected to receive a small amount of the respective token. Upon introducing the wallet address, the user is presented with a GotCHA challenge. To receive the tokens, the user must solve such challenge. We chose this setting as a prototype example because the authors have identified that currently most test network token faucets rely on centralized CAPTCHA providers as a solution mechanism. We believe that this is antithetical to the ethos of the system and highlight that there are examples of faucets that require users to sign with with personal accounts (e.g., GitHub, Gmail, ...) in order to be able to perform requests to such faucets. A space so called decentralized must not rely on such solutions.

A. Registration

To participate in the GotCHA challenge provisioning, a worker must register with an underlying smart contract. To perform this registration, the worker must call a registration function within the smart contract that basically receives a simple transaction to confirm that a worker is the owner of a specific wallet address. We highlight that this approach is currently vulnerable to sybil attacks. The necessary constraints to provide the necessary resistance against this type of attacks is beyong the purpose of this paper.

We have produced two smart contract implementations for this registration service. These implementations rely on two different platforms, namely the Internet Computer (ICP) [10] and Algorand [11]. The purpose of the registration service is to effectively act as a public-key infrastructure where anyone can freely join the network and/or check which workers are registered in the system.

B. Workers

Workers are developed in C# and run a GotCHA challenge database instance along with a web server—configured with TLS—exposing an endpoint able to be queried for GotCHA challenges. The worker implementation features a UI to allows users to quickly register with the smart contract using their (public) IP address and worker ID.

C. Web Server

a) Front end: The front end implementation, available in [12], simulates a faucet environment with a simple text field for the user to introduce a wallet address along with a button to initiate the transaction of the test network tokens. Upon clicking on the button, the user is presented with a GotCHA challenge. In the event the user would be unable to successfully complete the GotCHA challenge, the user would be redirected to another page. Thus, not receiving the tokens. It is beyond the scope of this work and, as of the time of writing, a CAPTCHA system to manage unsuccessful GotCHA challenge completions. From the author's perspective, it is the responsibility of the application making use of a CAPTCHA system to determine what happens when a user fails to complete a challenge. The nature of the service and content being protected by a CAPTCHA challenge play a role may require different policies for unsuccessful CAPTCHA challenge attempts.

b) Back-end: The back-end is comprised of a web server developed with Node.js that requests the available workers from the smart contract and relays the GotCHA requests to the worker that provides a GotCHA challenge. Additionally, the smart contract is deployed on the testnet for the Algorand platform and is developed in PyTeal. For the Internet computer blockchain, the smart contract is developed in Motoko and is deployed as a canister on a local canister network.

D. Future Work

Presently, the authors are working on a more complete, and production-ready implementation of GotCHA . This implementation features an extra server, responsible for the registration of web servers in the system with the use of blind signatures. As a result, web servers can join and use the network without the GotCHA platform knowing which entities are behind the web servers in the system, thus adding an additional privacy layer. Furthermore, this ensures that workers can successfully authenticate the web servers in the system when receiving requests for new challenges.

VIII. Previous Work

This section comprises a summary of the relevant previous work in the CAPTCHA space and purposefully does not describe any of the shortcomings in each of these references.

a) reCAPTCHA [6]: is a CAPTCHA solution that started by asking users to decipher hard to read text or match images. This approach, however, evolved to use an advanced risk analysis engine and adaptive challenges to keep malicious software from engaging in abusive activities on a website. The second version, reCAPTCHA v2, verifies if an interaction is legitimate with a checkbox and *invisible* badge challenges.

b) hCaptcha [13]: is a CAPTCHA service provider attemtpting to replace reCAPTCHA and is mainly designed for using human labor to label machine learning datasets for different companies. hCaptcha pays Human Tokens ($HMT) to the website owners hosting the hCaptcha service for every CAPTCHA challenge that is successfully solved by the corresponding visitors.

c) Cryptographic Attestation of Personhood [14]: is a solution by Cloudflare where the user, solves the provided challenge by clicking on the I am human button and gets prompted for a security device. User decides to use a Hardware Security Key [15], and taps said security key. A cryptographic attestation is then sent to Cloudflare, which allows the user in upon successful verification.

d) Privacy Pass [16]: introduces an anonymous user-authentication mechanism suitable for cases where a user is required to complete some proof-of-work (e.g. solving an internet challenge) to authenticate to a service. In short, the extension receives blindly signed 'passes' for each authentication and these passes can be used to bypass future challenge solutions using an anonymous redemption procedure. For example, Privacy Pass is supported by Cloudflare to enable users to redeem passes instead of having to solve CAPTCHAs to visit Cloudflare-protected websites.

e) SQUIGL-PIX [17]: is a type of CAPTCHA challenge where a user is instructed to read and understand an instruction, which demands the tracing of an object on one of three presented pictures. A user must understand what to trace, then find the corresponding object in the set of the three pictures and trace it. If the user traces the correct object, the challenge is considered complete.

f) ESP-PIX [18]: is a CAPTCHA challenge where instead of typing letters, a user proves as a human by recognizing what object is common in a set of images. This proposal is regarded as the first example of a CAPTCHA based on image recognition.

g) Worldcoin [19]: is a solution that leverages biometrics, cryptography, and incentives to create a scalable, privacy-preserving solution where users utilize a biometric device, called the Orb, which captures an image of a person's eyes and converts it into a short numeric code and ensures that each person can only register once.

h) GeeTest [20]: is an adaptive CAPTCHA solution that automatically collects and analyzes multiple distinct parameters, that include the network environment data, device attributes, biometric data, and many others. Based on these parameters, the GeeTest security engine evaluates the risk level of the visitor. Through advanced machine learning models, GeeTest accurately detects and blocks fraudulent bot traffic in real-time. If bot features are detected, then more data is collected through a challenge response to verify the identity of the visitor.

i) MTCaptcha [21]: is a smart CAPTCHA service that is GDPR and WCAG compliant, providing the confidence of privacy and accessibility. Adaptive invisible noCaptcha insures friction-less verification for real humans whilst hard on bots. And our multi user account management and automated regression test support are just some of the features build for the needs of the enterprise.

j) Proof-of-Humanity [22]: is a decentralized, AI-resistant, and economically incentivized approach that incurs in a complexity cost as users must submit a video containing speech along with an Ethereum address and a deposit. The profile, once submitted passes to a *Vouching Phase* until the users gets vouched from an already registered user. Other users can alternatively challenge particular registrations. Once the profile receives a vouch, it is then moved to registered and the deposit is returned.

k) rtCaptcha [23]: also referred to as Real-Time CAPTCHA , requires users to look into their built-in camera of their mobile phone while answering a randomly-selected question that appears within a CAPTCHA challenge on the screen of the device. Overall, this new approach requires requests to pass four tests: successful recognition of a challenge question from within a CAPTCHA challenge, response within a narrow time window that only humans can meet, and successful matches to both the legitimate user's pre-recorded image and voice.

l) Lorenzi et. al [24]: present a generalized methodology to transform existing images and apply various noise generation algorithms into variants that are resilient to attacks by adversarial AIs. This research shows how noise addition can serve as an effective method to solve the scalability problem of image CAPTCHAs and effectively foil Reverse Image Search and Computer Vision attacks.

IX. Discussion

a) Security Notes: The security of the construction relies on the fact that different workers provide different types of CAPTCHA challenges. Therefore, an adversary must have an AI that specializes in recognizing which type of challenge is provided along with the corresponding training to subvert such challenge. We believe this is a much harder setting for an adversary to overcome. We leave as future work, the formal proof of security and corresponding empirical results of our system against an adversarial AI trained to subvert different types of challenges.

b) Open Problems: Presently, the main outstanding existing open problem in our system is the ability to mitigate click farm settings where the CAPTCHA solving is outsourced to humans. We highlight, however, that the team started developing solutions to tackle this issue by introducing, in the CAPTCHA challenges, concrete cultural elements that are easily identifiable for locals, or even people temporarily visiting a specific region yet hard to recognize by click farm workers. These findings, however, remain to be fully explored and defined.

c) CAPTCHA ads: An additional side effect of our approach is that by decentralizing the CAPTCHA solving process with the use of an underlying blockchain, we introduce the following properties: transparency, verifiability, and immutability. Therefore, websites displaying CAPTCHA challenges from our platform can get concrete metrics regarding how many people are viewing and solving the corresponding challenges. A potential consequence of this design is that advertisers can take advantage of this underlying challenge provisioning layer and integrate ads at a challenge level, thus ensuring that they have reliable metrics regarding their ad campaigns.

X. Conclusion

Traditional CAPTCHA approaches have been deployed by Big Tech companies to combat automated bots, and while they have been successful to some degree in preventing malicious activities, their centralized nature has caused several issues such as data privacy and scalability.

In this work, we proposed a novel decentralized CAPTCHA platform which leverages distributed ledger technology for secure CAPTCHA generation and verification. Overall, the proposed decentralized CAPTCHA platform is an effective solution for preventing malicious bot activities without sacrificing user experience or privacy. Therefore, decentralized CAPTCHAs provide an effective and reliable alternative to traditional CAPTCHAs for combating malicious bots and protecting data privacy at the same time. This research has opened up a promising avenue for future work towards developing more advanced CAPTCHA-based security measures. We hope that this paper inspires further exploration on the topic of CAPTCHA security and decentralization technology.

Acknowledgements

[Redacted]

References

[1] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, 10 1950. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433

[2] J. R. Searle, "Minds, brains, and programs," *Behavioral and Brain Sciences*, vol. 3, pp. 417–424, 1980.

[3] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in Cryptology — EUROCRYPT 2003*, E. Biham, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 294–311.

[4] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[5] TechCrunch, "Google now using recaptcha to decode street view addresses." [Online]. Available: https://techcrunch.com/2012/03/29/google-now-using-recaptcha-to-decode-street-view-addresses/

[6] "recaptcha." [Online]. Available: https://www.google.com/recaptcha/about/

[7] M. Motoyama, K. Levchenko, C. Kanich, D. Mccoy, G. Voelker, and S. Savage, "Re: Captchas–understanding captcha-solving services in an economic context," 09 2010, pp. 435–462.

[8] B. News, "Pc stripper helps spam to spread." [Online]. Available: http://news.bbc.co.uk/2/hi/technology/7067962.stm

[9] hCaptcha, “Privacy policy,” 2021, last accessed 16 December 2022. [Online]. Available: https://www.hcaptcha.com/privacy

[10] I. Computer, “Internet computer website,” 2022, last accessed 16 December 2022. [Online]. Available: https://internetcomputer.org/

[11] Algorand, “Algorand website,” 2022, last accessed 16 December 2022. [Online]. Available: https://www.algorand.com/

[12] GotCHA, “Gotcha demo,” 2022, last accessed 16 December 2022. [Online]. Available: https://www.gotcha.land

[13] “hcaptcha.” [Online]. Available: https://www.hcaptcha.com/

[14] “Introducing cryptographic attestation of personhood.” [Online]. Available: https://blog.cloudflare.com/introducing-cryptographic-attestation-of-personhood/

[15] “Yubikey 5 series.” [Online]. Available: https://www.yubico.com/products/yubikey-5-overview/

[16] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, “Privacy pass: Bypassing internet challenges anonymously,” *Proceedings on Privacy Enhancing Technologies*, vol. 2018, pp. 164–180, 06 2018.

[17] “Squigl-pix.” [Online]. Available: http://server251.theory.cs.cmu.edu/cgi-bin/sq-pix

[18] “Esp-pix.” [Online]. Available: http://server251.theory.cs.cmu.edu/cgi-bin/esp-pix/esp-pix

[19] “Worldcoin.” [Online]. Available: https://worldcoin.org/

[20] “Geetest.” [Online]. Available: https://www.geetest.com/en/

[21] “Mtcaptcha.” [Online]. Available: https://www.mtcaptcha.com/

[22] “Proof of humanity.” [Online]. Available: https://www.proofofhumanity.id/

[23] E. Uzun, S. Chung, I. Essa, and W. Lee, “rtcaptcha: A real-time captcha based liveness detection system,” 02 2018.

[24] D. Lorenzi, E. Uzun, J. Vaidya, S. Sural, and V. Atluri, “Enhancing the security of image captchas through noise addition,” vol. 455, 05 2015, pp. 354–368.