Censorship-resistant Web Annotations Based on Ethereum and IPFS

João Santos[†], Nuno Santos[†], David Dias[‡]
[†]INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
[‡]Protocol Labs
joao.marques.santos@tecnico.ulisboa.pt,nuno.santos@inesc-id.pt,david@protocol.ai

CCS CONCEPTS

• **Social and professional topics** → *Technology and censorship*; • **Information systems** → *Web interfaces*; • **Security and privacy** → *Distributed systems security*;

KEYWORDS

Censorship resistance, Web annotations, Ethereum, IPFS

ACM Reference Format:
João Santos[†], Nuno Santos[†], David Dias[‡]. 2020. Censorship-resistant Web Annotations Based on, Ethereum and IPFS. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3341105.3374049

1 INTRODUCTION

Flooded by the propagation of false or biased news in the Web, people tend to resort to social networks to read posts from reliable sources, exchange commentaries with trustworthy parties, access first-hand content, or cross-check information that appears in news outlets. However, platform providers like Facebook or Twitter can ultimately decide about the contents exposed to each user. Anecdotal evidence suggests that such platform providers are prone to pressure by political or economical agents, and may be ideologically driven to hide messages or block certain users [2, 12] thereby impairing users' ability to freely access rightful information.

This paper presents DClaims, a system that provides a censorship-resistant distributed service for the exchange of messages over the Internet using web annotations in a user-friendly manner. Web annotations [14] are defined in a W3C standard and empower users browsing the Web to highlight text on web sites they visit, create sticky notes or comment parts of a web page, and share it with friends. In a typical scenario, a user visiting a news webpage article that shows portions of text highlighted by her friends and when she places her mouse over text highlights she sees comments made by the users about that text. Web annotations allow for the display of an overlay of data on top of the existing websites without changing the websites' original resources.

To provide open and censorship-free access to web annotations, DClaims is characterized by a fully decentralized architecture based on two building blocks. First, it uses the Ethereum [3] blockchain to keep a permanent, canonical record of all annotations made by the end-users ensuring that they receive updated information, unfiltered, and ordered. Because the Ethereum blockchain is very large and decentralized, it is very difficult to be controlled by any government authority or media outlet. Consequently, thanks to Ethereum, DClaims exhibits strong censorship resistance properties in giving worldwide access to web annotations.

Given that the storage costs in Ethereum are high, to help reduce the amount of data stored on the Ethereum blockchain, DClaims uses a second building block – the Inter-Planetary File System (IPFS) [1] – to store the web annotations themselves, delegating to Ethereum the job of recording only their respective IPFS links. IPFS is a decentralized peer-to-peer file system that relies on a network of IPFS nodes for storing file replicas on a local repository and making them available to the rest of the network. Given that the files are indexed by IPFS links, i.e., hashes of their content, IPFS provides strong file integrity assurances. Furthermore, since every access to a file by a remote client may result in the creation of a new file replica on a local or nearby IPFS node, the content stored on IPFS will tend to be replicated across a larger number of IPFS nodes and therefore become more difficult to be blocked by state-level adversaries. For instance, IPFS has been used as a tool against state censorship by granting access to Wikipedia to Turkish citizens, following the government's blockage of the website [10].

To further reduce the costs of Ethereum transactions and improve the system scalability, DClaims includes additional proxies, named *publishers*. Publishers help decrease the number of transactions so that Ethereum's 20 transaction per second limitation does not turn into a bottleneck in the system. Nevertheless, publishers are not considered to be part of the trusted computing base of our system and their misbehavior cannot affect the censorship-resistance properties offered by our system: DClaims is designed so as to prevent publisher misbehavior and spamming.

We implemented a prototype of DClaims. Our evaluation shows that our system performs well, and that the web browsing experience of end-users is not significantly affected. We also analyzed the costs of a full-scale deployment of DClaims using the activity level of Facebook's news pages as an estimate for expected demand which suggests that our system is economically viable. We refer the interested reader to an accompanying technical report that provides fully detailed description of the DClaims system [11].

2 ARCHITECTURE

In DClaims, a web annotation is handled by four actors: the *creator* is responsible for the creation of a new web annotation, the

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03.
https://doi.org/10.1145/3341105.3374049

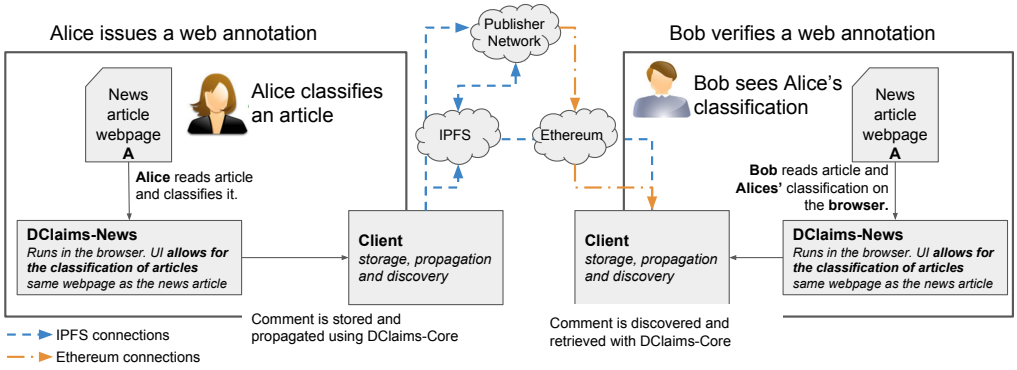

Figure 1: DClaims-News basic workflow: Alice writes a web annotation on a news article which is then read by Bob.

issuer submits the annotation to the backend, a *verifier* retrieves the annotation from the backend and verifies its authenticity, and a *viewer* displays the annotation on a web page. Figure 1 illustrates a web annotation workflow for a simple application, named DClaims-News, which lets users classify news articles with web annotations, in this case users Alice and Bob running DClaims-News on their browsers. When Alice visits a news website, DClaims-News instruments the article's web page to allow Alice to classify it as true, or false. DClaims-News (the creator) creates a new annotation based on Alice's choice, and forwards it to a publisher (the issuer), which submits the annotation to the backend. When Bob visits the same website, DClaims-News instruments that page on Bob's browser so that Bob can read attached classifications after the retrieval and verification of the article's annotations in background by the local client (the verifier) retrieves. DClaims-News (the viewer) displays the existing classifications of that article, including Alice's. We use this example to present the design details of DClaims.

Data structures: To manage the system's state, we use three main data structures: *claims*, *web annotation smart-contract*, and *publisher registry smart-contract*. Claims are used primarily (but not exclusively) to encapsulate web annotations provided by applications along with additional metadata, e.g., digital signatures and user identification. Claims are represented as Verifiable Claims [13], a W3C data specification standard for expressing rich sets of signed statements. The web annotation claim format can be further customized by the application developer, e.g., to represent simple *true* / *false* statements, structured records, text, images, etc. The web annotation smart-contract is an Ethereum smart-contract that keeps track of the claims issued and stored on IPFS by saving the respective claims' links (i.e., self-describing content hashes). For instance, for the DClaims-News application, this smart-contract holds a hash list where the key is named *topic*. The list contains the IPFS links, issuer addresses, and time stamps of all the claims about that topic (represented by an URL). Lastly, the publisher registry smart-contract is a second Ethereum smart-contract that maintains a directory of all publishers and keeps track of the complaint claims that might have been issued against potentially misbehaving publishers.

Claim management operations: In DClaims, the life cycle of a claim involves four main operations, namely: *creation*, *issuance*, *verification*, and *revocation*. Creation consists in the generation of a claim based on the input provided by the user. The content of the claim is signed by the claim creator (see DClaims-News in Figure 1) so that the the claim's validators can check its authenticity and integrity. The issuance operation consists in the submission of the claim to DClaim's backend resting upon IPFS and Ethereum. A claim can be issued by the client or by a publisher on behalf of the client. The issuer signs the claim with its private key, stores the claim on IPFS so as to obtain the respective IPFS link to the claim, and inserts the link into the hashlist of the web annotation smart-contract. The smart contract contains a record of the issuer ID which serves to identify the entity who paid for the transaction and issued the claim. The claim itself contains a second ID which identifies the user that has provided the enclosed web annotation. Claim verification is invoked by the application before displaying the annotations to the user. First, the client (Bob's client in Figure 1) queries the smart-contract to get the IPFS links of the claims associated with the queried topic (URL) and retrieves the respective files from IPFS. Then, all claims from issuers who are not white-listed by the user are discarded. Finally, after validating the signature of the claim issuer, and checking that the claim has not been revoked, the claim is considered to be valid and handed over to the application. It is also possible to revoke a claim by issuing a special revocation claim, which is tagged with revocation type, includes the UID of the claim to be revoked, and is preserved in the DClaims backend.

3 IMPLEMENTATION

We implemented DClaim's client library, the publisher software, and Ethereum smart-contracts. Client and publisher were written in Javascript. In the client, we used Go-IPFS [5] to connect to the IPFS network and the JS-IPFS-API [8] library to communicate with the Go-IPFS node running locally. The publisher code runs on a Node.js web server. We wrote the Ethereum smart-contracts in Solidity, which is similar to Javascript in syntax, but typed. To deploy smart-contracts on the Ethereum network, we run an Ethereum node using the Go-Ethereum [4] (Geth) client. For the DClaims client we used Metamask [9], which is a browser extension that acts like a gateway to Ethereum nodes maintained by Infura [6] (which runs public Ethereum nodes). Node.js and Javascript browser applications can connect and interact with an Ethereum node via the web3.js library [15].

We built DClaims-News, the web annotation application for three news websites. It allows users to classify and view classifications on news articles, and consists of a browser extension for

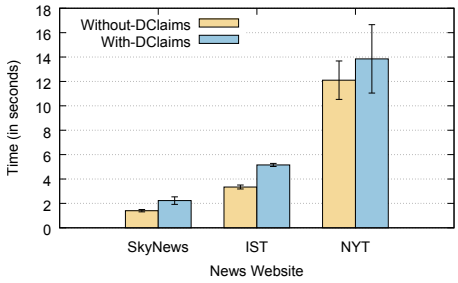

Figure 2: Full website loading times.

Chrome. It implements a visual overlay placed on top of the news websites. To draw the visual elements (e.g., buttons to interact with the application) we injected several Javascript files via a Chrome browser extension to change the HTML of the web pages. Claim issuance can be performed directly by the application or mediated by a publisher. When issuing a claim, even if using publishers, users must sign in using an Ethereum Wallet Address. To sign and verify signatures we leveraged Metamask.

4 EVALUATION

To assess whether DClaims can provide a viable alternative to current web commentary platforms, such as social networks, we evaluate the performance of the DClaims-News browser extension.

Performance of the Web client: We report the performance results of DClaim-News application. To assess the impact of DClaims to end-users' experience, we adapted our web extension to support three websites: SkyNews, New York Times (NYT), and Instituto Superior Técnico (IST) [7], which is the news front page of a university website. Figure 2 shows the time all three news websites take to load completely, with and without DClaims-News enabled. The overhead DClaims introduced varies between 0.4 and 2 seconds and corresponds to the time that the web extension needs to connect to both an IPFS and Ethereum nodes, and then, for each article, it needs to generate the news article ID (the SHA-3 hash of the article URL). Note that, since this operation takes place in background while the web page is being rendered to the user, this is not the latency perceived by the user. From our experience using DClaims-News, we did not perceive a significant degradation of performance, as the original website elements (e.g., news titles, images) are rendered as before; only the elements introduced by DClaims (view claims button, claims counter) take longer to appear.

Cost analysis: We analyze the costs of a full-scale deployment of DClaims, which essentially correspond to the costs of sustaining the publishers network. We start by estimating the level of activity to be handled, using Facebook data to model the potential workload. Next, we calculate the costs of publisher resources based on the considered activity level. Finally, we analyze the cost of the system, offering an example as to how it compares to real-world systems in use today. Table 1 presents the main findings of our study. Running the DClaims system for one big news outlet, such as CNN, Fox News, BBC News or The New York Times, would approximately cost USD 281152 per year. This value was calculated for only one of these large news outlets, so the value presented does not represent the real world cost. However, even if we assume that around the world there are 30 news outlets the size of the ones analyzed, DClaims' costs are still significantly smaller than the ones for real-world systems with a donation based financial model, such as Wikipedia.

Final Costs	
Storage	2203
Computation	1880
Ethereum	277069
Total cost for 1 year	281152
Useful Metrics	
Cost per 1000 Claims (USD)	2,54
Cost per User for 2,7M users (USD)	1,041

Table 1: Final cost analysis per news outlet

Put in perspective, these values are reasonable. The Facebook news pages analyzed have, on average, 27 million users. Even if we assume that DClaims only attracts 1% of those users the cost per user, per news outlet, would be less than USD 1 per year. DClaims targets users who need to circumvent censorship. Many people pay monthly fees for security services such as VPNs, ranging from USD 5 to 10 per month, which equals USD 60 to USD 120 per year. Therefore, if a donation based financial model such as Wikipedia does not succeed, there is reason to believe a subscription-based service, would. Thus, we infer there is both a market and a viable financial model for third-parties willing to host DClaim publishers.

5 CONCLUSION

This paper presents DClaims, a decentralized web annotations platform which is resistant to censorship. DClaims stores data in a distributed network and keeps a registry of metadata on the Ethereum blockchain, which is a tamper-proof, permanent record of information. To address the limitations of blockchain technology, DClaims uses a small network of dedicated nodes called publishers. We built a reference implementation of the system on the form of a browser extension, which allows for the web annotation of news websites, allowing users to classify news articles, and view the classifications made by others. Our evaluation shows DClaims can support the same level of activity of Facebook's news organizations pages.

Acknowledgments: We thank the anonymous reviewers for their comments and suggestions. This work was partially supported by national funds through Instituto Superior Técnico / Universidade de Lisboa and FCT via project UID/CEC/50021/2019.

REFERENCES

[1] Juan Benet. 2014. IPFS-content addressed, versioned, P2P file system. *arXiv.org* (2014).
[2] BusinessInsider. 2017. Twitter Has Gone From Bastion of Free Speech to Global Censor. (2017).
[3] Vitalik Buterin. 2013. Ethereum white paper. (2013).
[4] Go-Ethereum. 2018. (2018). https://github.com/ethereum/go-ethereum
[5] Go-IPFS. 2018. Go-IPFS. (2018). https://github.com/ipfs/go-ipfs
[6] Infura. 2018. (2018). https://infura.io
[7] Instituto Superior Técnico. 2018. (2018). https://tecnico.ulisboa.pt/en
[8] IPFS. 2018. JS-IPFS-API. (2018). https://github.com/ipfs/js-ipfs-api
[9] MetaMask. 2018. (2018). https://metamask.io
[10] Observer. 2017. Turkey Can't Block This Copy of Wikipedia. (2017). http://observer.com/2017/05/turkey-wikipedia-ipfs/
[11] João Santos, Nuno Santos, and David Dias. 2019. DClaims: A Censorship Resistant Web Annotations System using IPFS and Ethereum. *arXiv.org* (2019).
[12] TheVerge. 2018. Republican Lawmakers Keep Grilling Mark Zuckerberg About 'Censoring' Two Conservative Vloggers. (2018).
[13] W3C. 2018. (2018). https://www.w3.org/TR/verifiable-claims-data-model/
[14] W3C. 2018. W3C Web Annotations. (2018). https://www.w3.org/annotation/
[15] Web3 Library. 2018. (2018). https://github.com/ethereum/web3.js