CSCI 432 Project Report

Dan Kercher and Vivek Bigelow

**Introduction**

Stegastamp is a web application built to help visual artists prove they are the original creators of a digital image. Combining the techniques of steganography and timestamping, Stegastamp provides the simplicity, transparency, trust, and security necessary for authorship proofs.

**Problem**

With the advent of the internet and social media, it is easier than ever to publicly share images. Unfortunately, these same technologies have facilitated a parallel increase in content theft and misrepresentation. How does one prove that they are the original creator of an image posted publicly on the internet?

To be widely adopted, any successful process must be simple, transparent, trustworthy, and secure. If the process is not simple, content creators may not find it worth their time to secure images before posting them. Without transparency and reproducibility, it becomes difficult to disseminate authorship proofs on the large scale required of the internet. Without trust in the process and tools, authorship proofs will likely be met with skepticism. Finally, without security, the process will be susceptible to compromise and cannot function as a reliable proof system.

Several methods already exist to prove digital content authorship. Examples include emailing or physically mailing content to oneself, relying on file metadata or screenshots, and watermarking and steganography; however, each of these methods fails at least one of the above requirements. Regarding email/mail and metadata/screenshots, a digital forensics team may be able to prove the authenticity of these methods; however, it is difficult to prove that these methods were free from tampering when sending them to a wide audience over the internet. Likewise, watermarking and steganography by themselves can be removed by skilled users or specialized tools and do not provide the means to prove the creation date.

**Implementation**

Our group initially identified steganography, PGP encryption, and server timestamping as a sufficient combination of tools to solve the authorship proof problem and began developing a python app to implement these strategies. After additional research, we decided that PGP was complicating the process without adding any benefit and that a web application would increase our accessibility and simplicity.

Our finished project, named Stegastamp (STEGAnography + timeSTAMP), is more of a guide or collection of tools than one single product. Our site guides content creators through a simple three-step process to ensure their authorship is provable in the future. To set up the process for a newly created image, authors simply follow these three steps:

1. Upload an image you created
2. Type in your name and the license you wish to assign to the work and save the new file
3. Upload the newly saved file to a trusted timestamp server and save the corresponding .ots file with the image

To prove authorship of an image, similar steps are provided:

1. Upload a file that has been stamped
2. The site will show the author and license information (optionally requiring a passphrase)
3. Upload the file's corresponding .ots file to the timestamp server
4. Upload the message-encrypted file to the timestamp server, and the combination will return a stamp date

Step 2 of the above processes implements the steganography portion of our solution. An opensource JavaScript steganography library is used to encrypt author information client-side. This version of the steganography process is done entirely in the user's browser and does not require sending the original image to a server for processing. The user's message (name and license) is first encrypted with AES-128 and a HTML 5 Canvas breaks the image down into color arrays listing the RGB+alpha colors of each pixel. The message is then encrypted into the color arrays. If the author intends to make the proofs public, it is not advised to set a message passphrase; however, if they would like to keep their proof private the message is encrypted into the image at a random start position determined by a sha256 hash of their message passphrase.

Once the steganography process is complete, step 3 uses a trusted timestamping server to generate an .ots timestamp file for the image. Like the steganography library, the timestamping service is completely opensource and all hashes are done client-side. Image files are read and a sha256 hash of the file is created in the user's browser. This hash is then collected in a batch to be pushed onto a Bitcoin blockchain and the user is given an .ots file based on their uploaded image to verify their stamp later. Periodically (usually with a delay of 1-2 hours), the server will push the batch of hashes to the chain, saving the block head to several calendar servers, and allowing the stamp to be verified. After uploading an .ots file, the service checks the calendar servers for the corresponding Bitcoin block header, then finds the path to the timestamp itself and returns the date of the push. Since hashes are pushed to the chain in delayed batches, the timestamp returns a date instead of an exact time.

**Test Results (Refer to "tests" folder included with the project)**

Follow the steps for "Stegastamp Proof Steps" in the instruction section of the site. If you upload "noPasswordDemo" from the StegaStamped folder, the author and license information should show "Dylan Schwab CC BY" If you then follow steps 3 and 4 to the Opentimestanmps.org site, you can upload "noPasswordDemo.png.ots" followed by "noPasswordDeno" to show that Bitcoin block 326733 attests that this file was stamped on 4/19/2020. You can repeat these same steps with "spiderPasswordDemo" and "spiderPasswordDemo.png.ots" using the password "spider"

Finally, a realistic scenario is considered for demonstration purposes. Imagine that Dylan Schwab created the file "overrideTestOriginal" and stamped it as "overrideTestOriginal.png.ots"

before posting it to the internet. I, Dan Kercher, then re-post the image later, claiming to be the original author. Dylan provides the .ots file publicly to prove that he is the original author. First, I find that if I, Dan, try to upload "overrideTestOriginal" I will not be given the option to add a message. If, however, I change the image, manage to remove the encrypted message, or somehow get a copy of the original image, I can add myself as the author (as was done in "overrideTestSecondStamp") Unfortunately, if I then timestamp my new file, Dylan will have an older stamp and more legitimate claim. I then get the idea to use his original .ots file with my overridden author message; however, you will find that if you try to verify "overrideTestSeconfStamp" with the original "overrideTestOriginal.png.ots" file, the server will return that the file does not match the original. This also demonstrates that timestamping is not enough on its own since the possession of an .ots file should not be left to determine authorship by itself. Ultimately, Dylan has a more legitimate claim as the original author since the oldest stamp is linked to a file with their name in the author message.

**Summary/Future work**

By combining the processes of steganography and timestamping, we were able to accommodate each of our goals of simplicity, transparency, trust, and security. The web application and simple step-by-step instructions make it simple for content creators to secure and later prove authorship. The use of opensource tools (given to users in the about section of the site) provides the transparency necessary to be reproducible and create/send proofs on a large scale. Regarding trust, the timestamp server provides a JavaScript library to stamp and send files directly to their servers; however, it felt like this diminished trust. Timestamping, much like certificates, is extremely reliant on the trustworthiness of the issuing authority. If you are not stamping files directly on the authority's site, you lose the benefit of the authority's reputation. Finally, security is maintained in multiple ways, including client-side tools, optional steganography passphrases, and the immutability of blockchain technology.

Future work on this project would first be to host the site. Additional steps would be to add more information to the site to continue the theme of it being an information hub. We already included links for users to learn about the opensource tools used, as well as different creative license; however, we could also include more information about how the tools work, and additional steps they can take to secure their work. Finally, we could expand the instructions section to include advanced instructions explaining the difference between passphrase or public steganography and the pros and cons of each approach.

**References (links to open source tools and resources, licenses can be found in the project folder)**


- https://opentimestamps.org/
- https://github.com/oakes/PixelJihad
- https://startbootstrap.com/themes/grayscale/.
- All test images courtesy of a friend, Dylan Schwab