

6-13-2018

# Authenticated Authorship

Andrew Leonard

Santa Clara University, aleonard@scu.edu

Mikhail Smelik

Santa Clara University, msmelik@scu.edu

Stephen Chuang

Santa Clara University, schuang@scu.edu

Follow this and additional works at: [https://scholarcommons.scu.edu/cseng\\_senior](https://scholarcommons.scu.edu/cseng_senior)



Part of the [Computer Engineering Commons](#)

---

## Recommended Citation

Leonard, Andrew; Smelik, Mikhail; and Chuang, Stephen, "Authenticated Authorship" (2018). *Computer Engineering Senior Theses*. 100.

[https://scholarcommons.scu.edu/cseng\\_senior/100](https://scholarcommons.scu.edu/cseng_senior/100)

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Computer Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact [rsroggin@scu.edu](mailto:rsroggin@scu.edu).

**SANTA CLARA UNIVERSITY**  
**DEPARTMENT OF COMPUTER ENGINEERING**

Date: June 13, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

**Andrew Leonard**  
**Mikhail Smelik**  
**Stephen Chuang**

ENTITLED

**Authenticated Authorship**

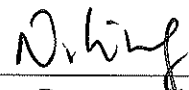
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE ENGINEERING

&  
^



Thesis Advisor



Department Chair

6/13/18

# **Authenticated Authorship**

by

Andrew Leonard  
Mikhail Smelik  
Stephen Chuang

Submitted in partial fulfillment of the requirements  
for the degree of  
Bachelor of Science in Computer Science Engineering  
School of Engineering  
Santa Clara University

Santa Clara, California  
June 13, 2018

# **Authenticated Authorship**

Andrew Leonard  
Mikhail Smelik  
Stephen Chuang

Department of Computer Engineering  
Santa Clara University  
June 13, 2018

## **ABSTRACT**

Recently there has been an increase of fake news which makes it so that people have a more difficult time of understanding if the article that they are reading is real or not. In addition, it is significantly easier for people to fabricate an article and pose it as if it was created by another person. Our solution to these problems is to create a system which will allow authors to sign their documents and publish them with a generated signature. This signature will allow readers to see if the article has not been tampered with as well as find out who the author of the article is by checking if the name given by the signature matches the author's name.

## **Acknowledgments**

Dr. Ahmed Amer, our Senior Capstone Advisor, for all his help and guidance with our project.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Requirements</b>	<b>3</b>
2.1	Functional Requirements . . . . .	3
2.2	Non-Functional Requirements . . . . .	3
2.3	Design Constraints . . . . .	3
<b>3</b>	<b>Use Cases</b>	<b>4</b>
3.1	Author . . . . .	4
3.2	Reader . . . . .	11
<b>4</b>	<b>Architectural Diagram</b>	<b>16</b>
<b>5</b>	<b>Technologies Used</b>	<b>18</b>
<b>6</b>	<b>Design Rationale</b>	<b>19</b>
6.1	Justifications for UI . . . . .	19
6.2	Justifications for Technology . . . . .	19
6.3	Justifications for Technology Not Used . . . . .	19
<b>7</b>	<b>Obstacles Overcome</b>	<b>21</b>
<b>8</b>	<b>Societal Issues</b>	<b>22</b>
8.1	Social . . . . .	22
8.2	Political . . . . .	22
8.3	Ethical . . . . .	23
8.4	Sustainability . . . . .	23
8.5	Usability . . . . .	23
8.6	Environmental Impact . . . . .	23
8.7	Economic . . . . .	23
8.8	Manufacturability . . . . .	24
8.9	Health and Safety . . . . .	24
8.10	Lifelong learning . . . . .	24
8.11	Compassion . . . . .	24
<b>9</b>	<b>Conclusion</b>	<b>25</b>
<b>10</b>	<b>References</b>	<b>26</b>
<b>11</b>	<b>Appendix</b>	<b>27</b>
11.1	Atom Install Guide . . . . .	27
11.2	Keybase Account Creation Guide . . . . .	27
11.3	Atom Use Guide . . . . .	29
11.3.1	Authenticating . . . . .	29

11.3.2	Store Private Key to File	30
11.3.3	File Authenticating	30
11.3.4	Verifying	30
11.3.5	Sharing	30
11.4	Chrome Extension Installation Guide	31
11.4.1	Enabling Chrome Extension Developer Mode	31
11.4.2	Downloading the Authenticated Authorship Extension	31
11.4.3	Loading the Authenticated Authorship Extension	31
11.5	Chrome Extension Use Guide	31

# List of Figures

3.1	Article in Atom . . . . .	4
3.2	Authenticated Authorship Login . . . . .	5
3.3	Signed Article . . . . .	6
3.4	Article Verification . . . . .	7
3.5	Failed Verification - Altered Article . . . . .	7
3.6	Save Private Key to File . . . . .	8
3.7	Signing an Article Using a Private Key File . . . . .	9
3.8	Signing Article to HTML . . . . .	10
3.9	Text Article . . . . .	11
3.10	Text Article - Full . . . . .	12
3.11	Chrome Plugin - Verified Article . . . . .	13
3.12	Keybase.io User Page . . . . .	14
3.13	Chrome Plugin - Failed Verification . . . . .	14
4.1	Architecture . . . . .	16
11.1	Private Key Retrieval Step 1 . . . . .	28
11.2	Private Key Retrieval Step 2 . . . . .	28
11.3	Private Key Retrieval Step 3 . . . . .	29



# Chapter 1

## Introduction

Recently there has been an influx of fake news in the world. Also people's identities have been stolen on the Internet and used to write falsified articles under that person's name creating the possibility of destroying the person's reputation. Due to both of these problems people are no longer as confident in the media as they have been previously. The issue for the public is that they no longer know if the media that they are viewing is real or not. Also they cannot be confident that the article which they are reading has not been modified since it was posted or published on the Internet. We believe that people should have the opportunity to know who takes responsibility for an article on the Internet and also if that article has been modified or forged in any way. We plan to accomplish this by creating a system which links a special key with an entity online where by knowing the special key you will be able to know if the article was signed by the entity and if the article has been modified or not. For this we need a way to manage keys and also a distributed ledger on which to store the special keys.

While there are other solutions available right now that get close to solving this problem on the Internet, none of them incorporate all aspects that are required in one area and are not simple enough to use by the general public. One solution is the one created by the Trust Project. The Trust Project provides identifiers that can be used to judge if the article has been falsified or modified and who it has been written by. The issue with this solution is that even though it does provide data for users to analyze not all users will be able to understand the data and know if an article has been modified or that the author is who they claim to be. The issue with the Trust Project is that it does not provide a simple way verify articles and authors. The problem with managing keys can be solved multiple ways. Some of the ways are sharing the keys with everyone who you know will read your information by posting it on the website or sending it by email. This method works, however, the problems that arise with this method are that people need to know how to use encryption and decryption to be able to use this solution. The other issue is that this limits the amount of people who can be reading the article as only people with the key will be able to do this. There is a solution to this problem which gets close to satisfying all the issues of the problem which we are trying to solve and that is [Keybase.io](#)<sup>[5]</sup>. They are a key management service which ties the keys to the Bitcoin blockchain by using a Merkle tree<sup>[4]</sup>. By doing this they are able to provide a way to manage keys and also have them on a public ledger which allows any person to get anybodies keys and also verify who owns the key. They also provide the ability to sign and verify documents. However, this is the part where they have an issue as the way to sign and verify articles are limited to their website and using the terminal on a computer. This posses the issue of accessibility as it does not allow a simple and efficient way to use their service. To be able to sign and verify some text a person has to either always copy the text that they need and paste it into their website or know how to use the terminal. This limits the amount of people who would use this solution and therefore makes it that this solution is not popular or accessible for people.

Our solution is a proof of concept and it demonstrates how to use previous solutions to create a modular authentication system. Our system allows people to sign articles using a private/public key pair which is linked to the Bitcoin blockchain. Our solution also allows people to verify article and see if there has been any article or metadata modifications. Finally our solution is as secure as current PGP encryption is and as secure as the blockchain fabric is. To demonstrate the ability to sign articles we modified a text editor to have the capabilities of signing articles. To

implement verification we modified the text editor and also created a web plugin to make verification on Internet webpages simple. As the key manager which links to a public ledger we used Keybase. Our system demonstrates the possibility of implementing such a system in news rooms and also social media websites by modifying their text editors to function like ours and also the ability to enable instant verification on their websites or a plugin which will do the verification. Our system can be used by any person and allows the users to implement only the aspects of the system they need.

## Chapter 2

# Requirements

### 2.1 Functional Requirements

- The system must convert articles and private keys into encrypted signatures.
- The system must verify the correct articles and not verify the rest.

### 2.2 Non-Functional Requirements

- The system must be simple to use.
- The system must be easy to learn.

### 2.3 Design Constraints

- The system must work on operating systems that authors will likely use (Windows, Mac, Linux).
- The system must work on a web browser that readers will likely use (Google Chrome).
- The system must work on a minimum of one popular content sharing service and one website that the client chooses.

# Chapter 3

## Use Cases

### 3.1 Author

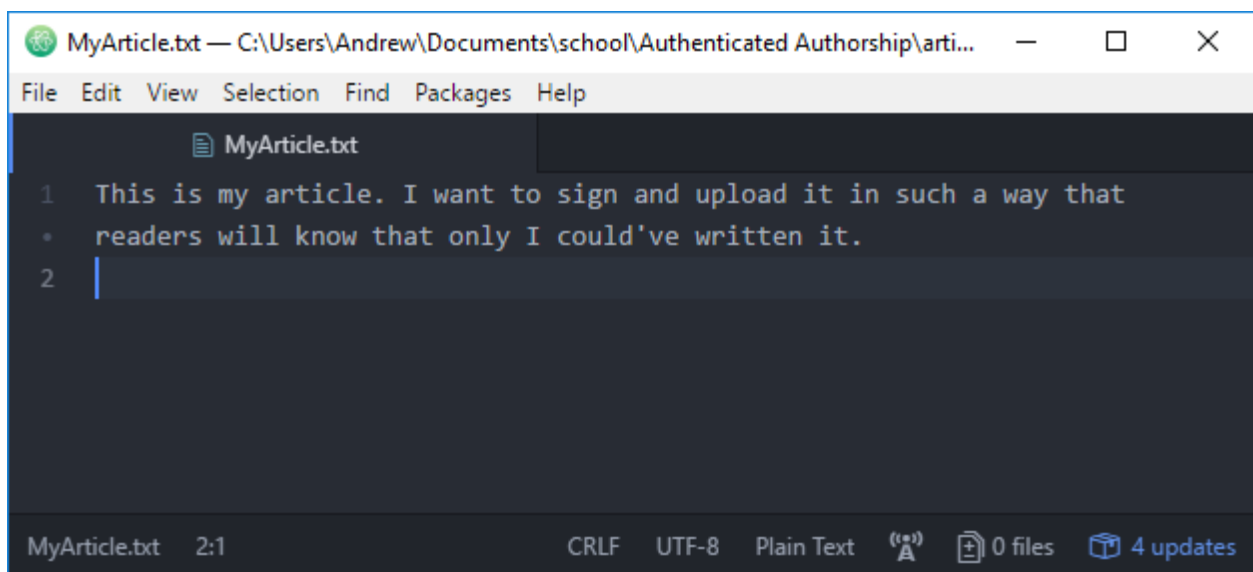


Figure 3.1: Article in Atom

Figure 3.1 shows an article written in the Atom text editor, ready to be signed using Authenticated Authorship.

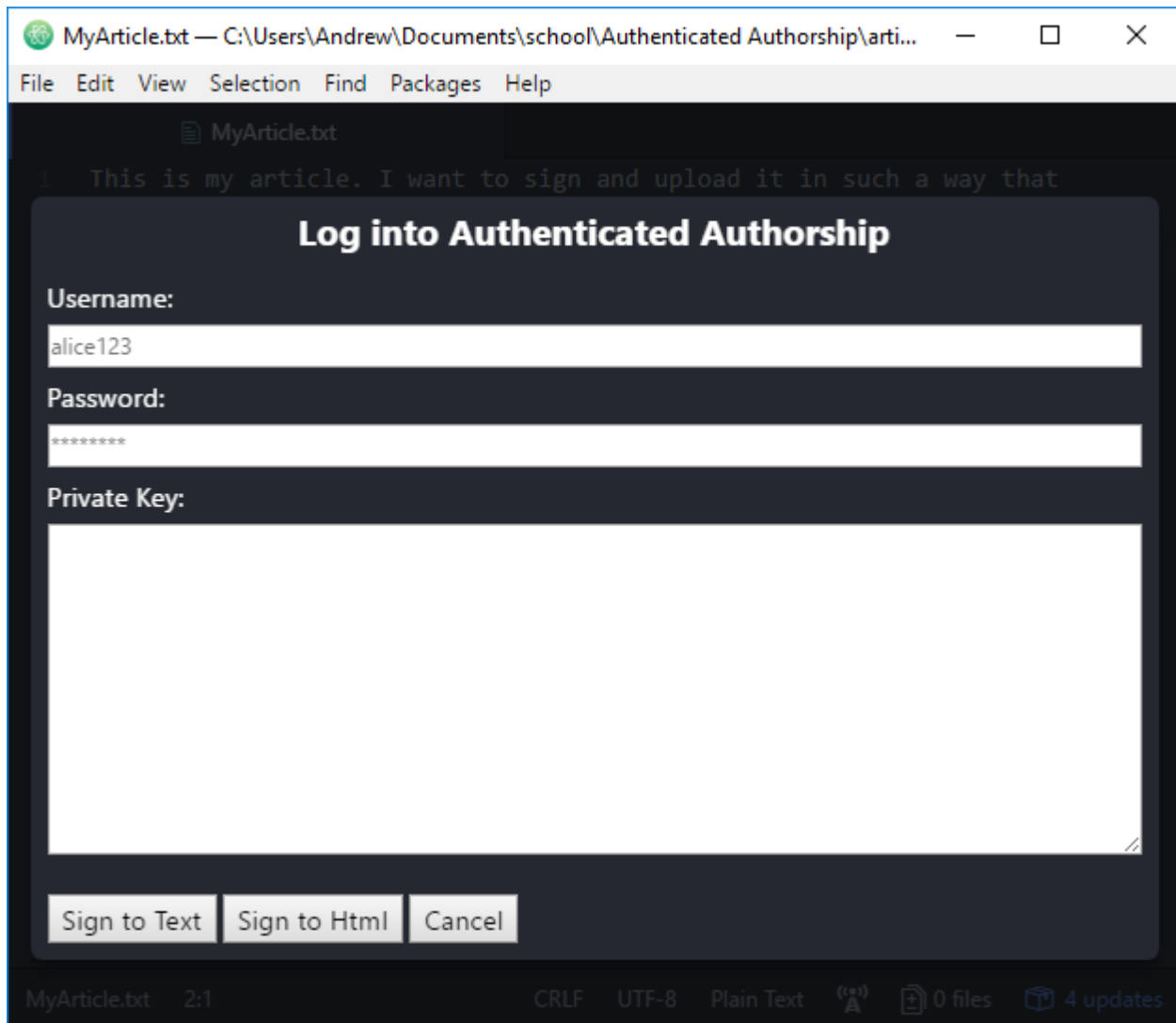


Figure 3.2: Authenticated Authorship Login

Figure 3.2 shows the login screen for the Authenticated Authorship Atom plugin, which asks for a username, password, and the user's keybase.io private key.

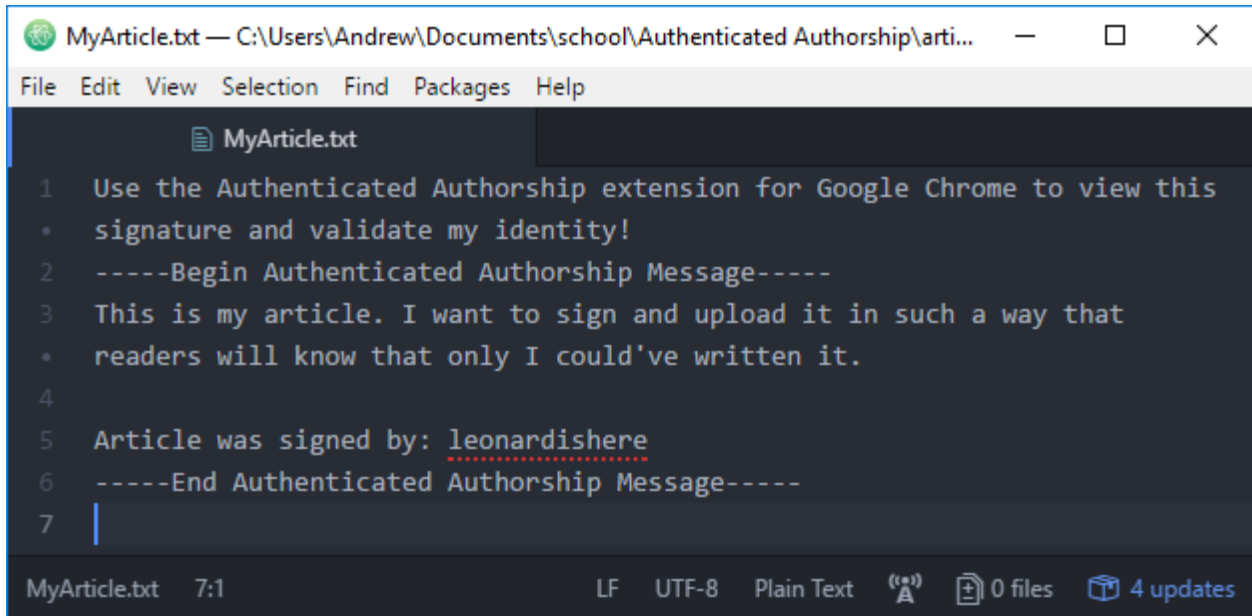
```
MyArticle.txt — C:\Users\Andrew\Documents\school\Authenticated Authorship\arti...
File Edit View Selection Find Packages Help

MyArticle.txt
1 Use the Authenticated Authorship extension for Google Chrome to view this
  • signature and validate my identity!
2 -----Begin Authenticated Authorship Message-----
3 This is my article. I want to sign and upload it in such a way that
  • readers will know that only I could've written it.
4
5 Author: leonardishere
6 Signature:
  • LS0tLS1CRUdJTiBQR1AgTUVTU0FHRS0tLS0tClZlcnNpb246IEtleWJhc2UgT3B1b1BHUCB2Mi
  • wLjc3CkNvbWw1bnQ6IGh0dHBzOi8va2V5YmFzZS5pby9jcnldG8KcnlnQ3VBbmljQVdJQm5mN
  • VEUU1BQ2dFbllVUFJswURORgdITE1uVUFXd1hiWms4MWJsZFZSVVV5U25salV5OHgKYzJKaVVG
  • EpOM05XZDJoTVR5SkdjblpGVTAxU1ZsRTJUVFprU0djOXdzQmNCQUFCQ2dBR0JRSmJCZHRtQUF
  • SgpFQ2RoUTlHVmdNME9YYzBILzF6ZXhSbWxDOWRjdWExcw1BSm4wZ285QVFzSGcxZmhLRlFDbn
  • jclhZQUJCQzBDcklxYmtPQytuY0wzay8rLzlfRmhrZW9GZFdUUKVscHpRNWtpRTRvaTMzZEZzW
  • JSOUo5d0EzVDU0V01RV0FVRlEKY2xyK0gzTFBvNl00QmRSbkFrN08xR1MxZ3NiNUdITURMTWQ3
  • ldFamd4ZjdhWFYmJmZvcwZFRzQ0RmFRUw1qRgpUNE1wL0hTVkJoeGxxM0pnaHlZR2w4TzcxUlB
  • dzJvWlFQWWhYOXV2c2Y4TXlhQ0NuUHdsRjY4ZURJbGhIbi9sCnh3Uk1ERDNITTd4WHM5WnZlUm
  • uMmtpeS9MVTFNRXZ5SzVOTWx4T0hmb1FxYkQ3a0pLVzRETYttbUkwRHh3Wk8KNVhER2duTytxe
  • J6MfdUa0pLNmV10XRyOHdLUDhJUEFSckZGaG9ib3RKbmUKPUNMTnkKLS0tLS1FTkQgUEdQIE1F
  • 1NBR0UtLS0tLQo=
7 Hash: 05nWUEE2JycS/1sbbPYI7sVwhLNRFrvesMRVQ6M6dHg=
8 Version: 1.0.0
9 -----End Authenticated Authorship Message-----
10

MyArticle.txt 10:1 LF UTF-8 Plain Text 0 files 4 updates
```

Figure 3.3: Signed Article

Figure 3.3 shows the results of logging in to Authenticated Authorship and signing the article to HTML. The output contains the original article, as well as metadata including the author's keybase username, the article's signature, and the hash for the article and signature. This text can be copied and pasted onto any website, and used with the verification function of both the Atom and Google Chrome plugins.



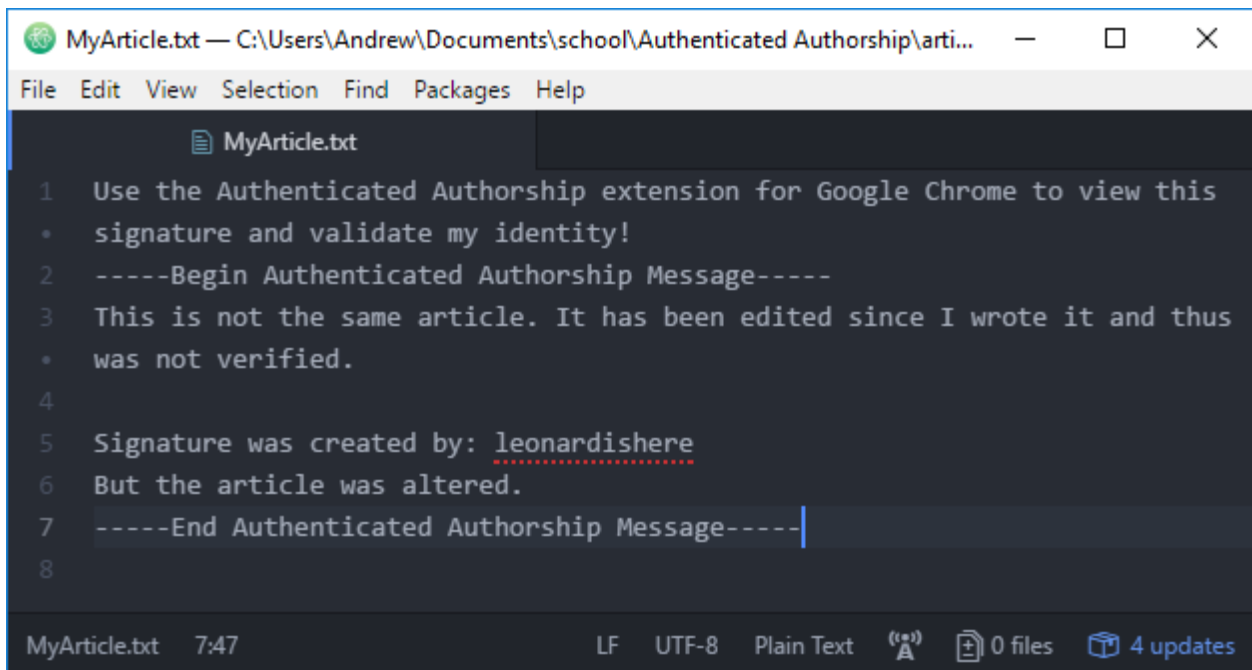
The screenshot shows the Atom editor interface with a file named 'MyArticle.txt' open. The text in the editor is as follows:

```
1 Use the Authenticated Authorship extension for Google Chrome to view this
• signature and validate my identity!
2 -----Begin Authenticated Authorship Message-----
3 This is my article. I want to sign and upload it in such a way that
• readers will know that only I could've written it.
4
5 Article was signed by: leonardishere
6 -----End Authenticated Authorship Message-----
7
```

The status bar at the bottom indicates 'MyArticle.txt 7:1', 'LF', 'UTF-8', 'Plain Text', and '4 updates'.

Figure 3.4: Article Verification

Figure 3.4 shows the result of verifying an article in Atom that is signed through Authenticated Authorship plugin. A successful verification shows only the article and the author's name, removing the rest of the metadata from the text.



The screenshot shows the Atom editor interface with a file named 'MyArticle.txt' open. The text in the editor is as follows:

```
1 Use the Authenticated Authorship extension for Google Chrome to view this
• signature and validate my identity!
2 -----Begin Authenticated Authorship Message-----
3 This is not the same article. It has been edited since I wrote it and thus
• was not verified.
4
5 Signature was created by: leonardishere
6 But the article was altered.
7 -----End Authenticated Authorship Message-----
8
```

The status bar at the bottom indicates 'MyArticle.txt 7:47', 'LF', 'UTF-8', 'Plain Text', and '4 updates'.

Figure 3.5: Failed Verification - Altered Article

Figure 3.5 shows the result of a failed verification due to an alteration of text in the article after it was signed. The metadata is still removed except for the author, but the verification system adds a note signifying that the article was not properly verified.

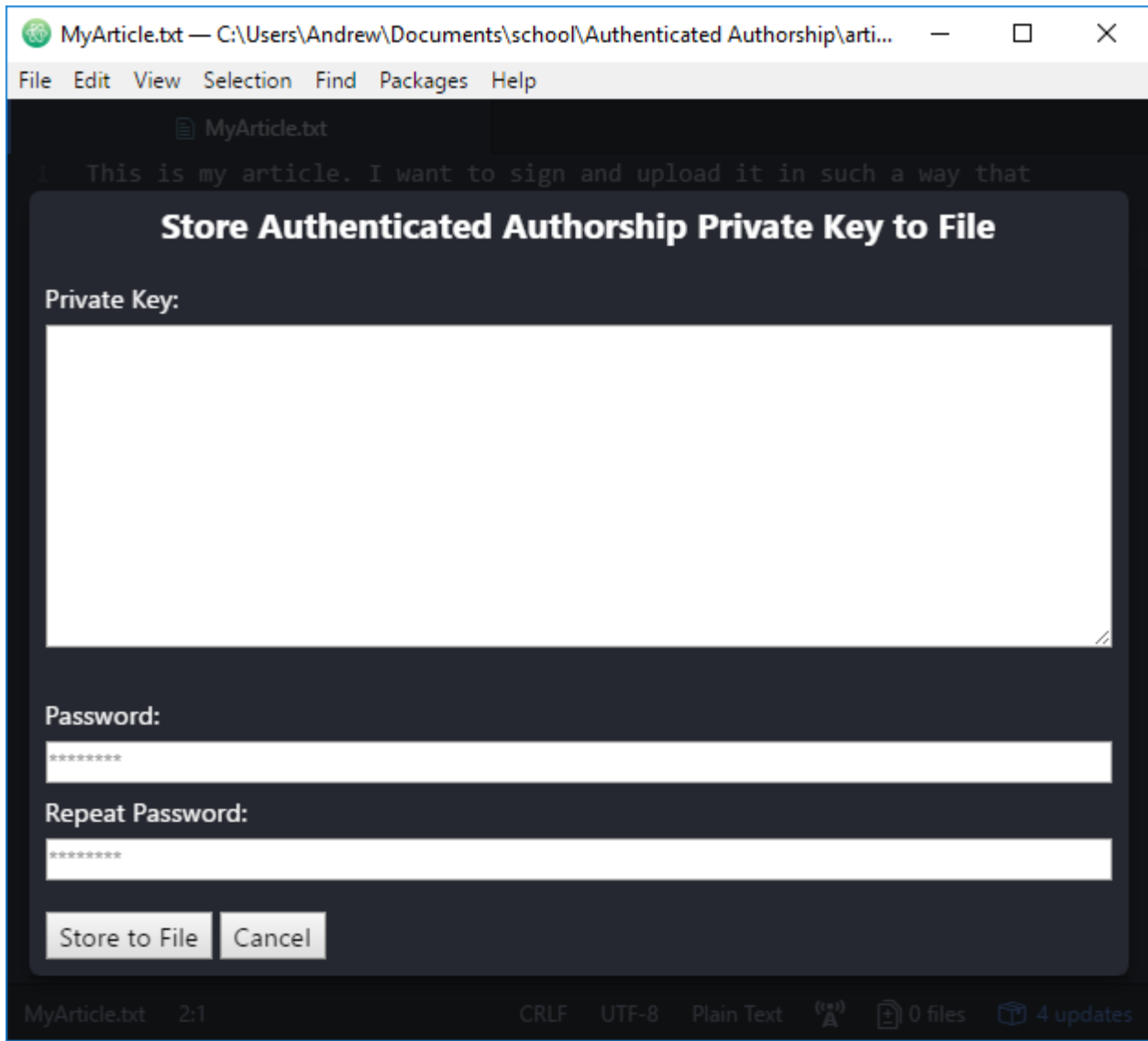


Figure 3.6: Save Private Key to File

Figure 3.6 shows the screen for saving a user’s private key to a file for ease of access, for people who frequently write articles and need to sign them. When a private key and a password to encrypt the file are provided, the Authenticated Authorship plugin generates a file with the extension .aak, short for ”Authenticated Authorship key”, that the user can save in a convenient location.



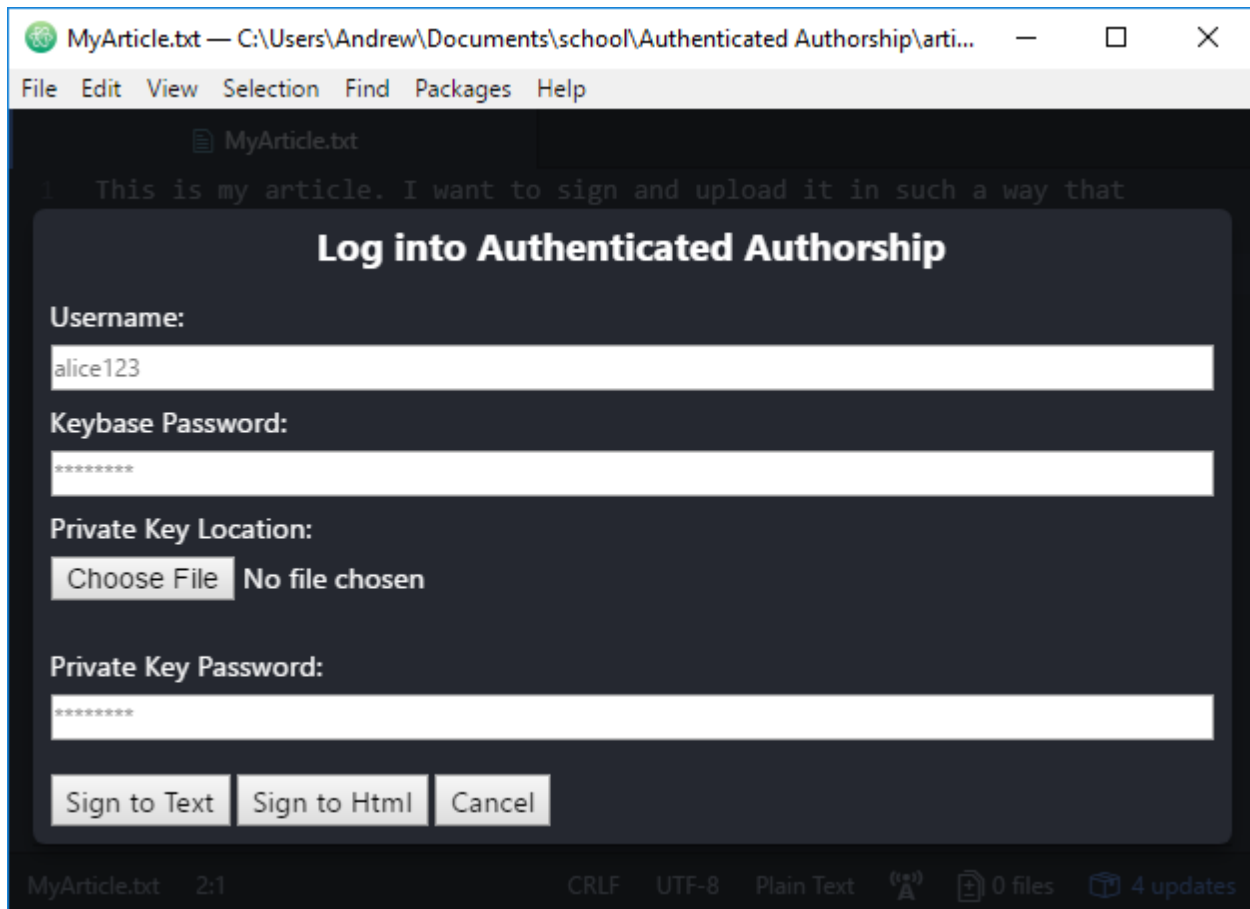


Figure 3.7: Signing an Article Using a Private Key File

Figure 3.7 shows the menu for signing an article using a private key file rather than manually entering the private key. Instead of entering the full private key, the user provides the .aak file as well as the password used to encrypt the file, eliminating the need to retrieve the private key from keybase.io every time an article needs to be signed.

The screenshot shows a code editor window titled "MyArticle.html" with the following content:

```

1 <article class="authenticated_authorship" data-author="leonardishere"
  • data-signature="LS0tLS1CRUdJTiBQR1AgTUVTU0FHRS0tLS0tClZlcnNpb246IEtleWJhc2
  • gT3BlblBHUCB2Mi4wLjc3CkNvbW11bnQ6IGh0dHBzOi8va2V5YmFzZS5pby9jcnlwdG8KCnlnQ
  • VBbmljQvdJQm5mN0VEU1BQ2dFbllVUFJ5WUR0RGdITE1uVUFxd1hkbW54MWFJZS5pby9jcnlwdG8KCnlnQ
  • 1V5OHgkYzJKaVVGbEpOM05XZDJoTVR5Skdjb1pGVTAxU1ZsRTJUVFprU0djOXdzQmNCQUFCQ2d
  • R0JRSmJCZDJhQUFvSgpFQ2RoUTlHVmdNME9SNF1IL211WGw2TWwVWVppa21CZHUuMHBVeD
  • 6bGtvRlpPWNpNFVkl5VzlpWHZRRXpClIxc0pXc1VQMhdacGJ5Qk10SnQzdUpjNkRiN0VDRUdiM
  • dJd1J1bTVLanVtd0d4eURQUU1YR2tDK2M1NExZL1gKejNiSXdQYUQrakROODgrcmRLRmkyUlJU
  • WdSdnJwTWQyb2NLYlpxTDVrNlJ1JUGV0QXpXRjBtOFV4cmt6MEhqSQpZT1FaQ1ZaQ1A3cCtUc2k
  • b09XMDkvM3NqZzVtTG4vMmpMcjYwSF1WY1JIRE50WFVPRVfmRjJ3dklabGhQajRTC1V6YkJDaw
  • qSXUwaUdPKzRRreE16Y2J4S0VLV2dDY0NlWkp1bldQNUFZM2xtOGJsSjc0ZmF6NG1FZ1VsVzBaM
  • IKN1BhN256cUFoTTRsRGNWd3VtVkZlZ2hCa1JRQk41OGFteVFDYVExcy9acGsKPUhFN3IKLS0t
  • S1FTkQgUEdQIE1FU1NBR0UtLS0tLQo="
  • data-hash="05nWUEE2JycS/1sbbPYI7sVwhLNRFrVESMRVQ6M6dHg="
  • data-version="1.0.0">
2   <p>This is my article. I want to sign and upload it in such a way that
  •   readers will know that only I could've written it.</p>
3 </article>
4

```

The status bar at the bottom of the editor shows "MyArticle.html 4:1", "LF UTF-8 HTML", and "0 files 4 updates".

Figure 3.8: Signing Article to HTML

Figure 3.8 shows the result of signing an article to HTML rather than text. By clicking the "Sign to HTML" button, the plugin generates valid HTML code that hides the metadata from the viewer but still makes it accessible for the plugin to verify. This code can then be pasted onto any website, and only the article will be visible on the page.

## 3.2 Reader

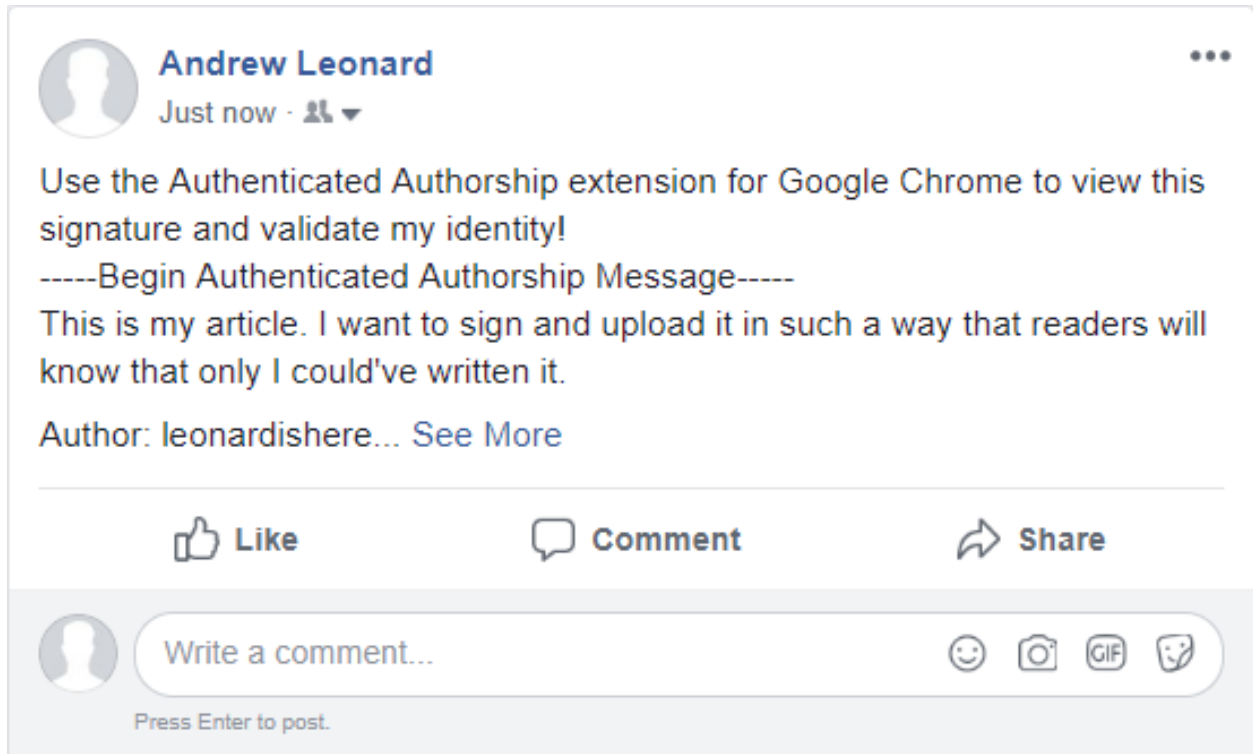


Figure 3.9: Text Article

Figure 3.9 shows an article signed by the Authenticated Authorship plugin, that is posted on Facebook. Without the Google Chrome plugin, the text will stay as is, with the metadata listed at the bottom of the post.



**Andrew Leonard**

1 min ·



Use the Authenticated Authorship extension for Google Chrome to view this signature and validate my identity!

-----Begin Authenticated Authorship Message-----

This is my article. I want to sign and upload it in such a way that readers will know that only I could've written it.

Author: leonardishere

Signature:

LS0tLS1CRUdJTlBQR1AgTUVTU0FHRS0tLS0tCIZlcnNpb246IEtleWJhc2Ug  
T3BlbIBHUCB2Mi4wLjc3CkNvbW1lbnQ6IGh0dHBzOi8va2V5YmFzZS5pby9  
jcnlwdG8KCnlnQ3VBbmljQVdJQm5mN0VEUU1BQ2dFblVUFJsWURORG  
dITE1uVUFXd1hpRzA4MWJsZfZSVV5U25saV5OHgKYzJKaVVGbEpOM  
05XZDJoTVRrSkdjbGpGVTAxU1ZsRTJUVFprU0djOXdzQmNCQUFCQ2dBR  
0JRSmJCZUliQUFvSgpFQ2RoUTIHVmdNME9oTWdlL0FuNERwT05KbDV  
NT1puMHZSNUZDWjRmMFNIMXI2YXpqYUEwei9qc3d2aGpxTHZ6CkEwZ  
ndrcDN6QkNYMndGR2R3VzJkTmVwVUFUTWdDYUk2UkJ6VmN1cWUySG  
pyekYvbjEwZXR0dIzEQzFTaWRHNzYKYUxhVkh3bDVFc2RJOVJ6TXIQME  
EvVVBuakR5dXZJdFhGMzUzVWZ1NnZmL1p1MlhtbmpKVmlRUXBNczU2  
N1B2cAo3STgybjI4THJhL0VqeTg5dXpqSjd3Y1FaNVRhTUwWcDFnMjdOV  
GFjWTY1V29XTU1sYzBKaGcvOU1XNk82L1pBCldMZkVhL3IKRmFkSkpWa  
lhWVGpiZ1RpT09JWmpKRWtLRkYwb2NqV0dEUWc3eVUrdkITNDZnRThM  
STZteUZTUIQKUU9VRksrTEdwL0pUWIZwQjVWNmJRaUVDdnRUZFc3Um  
hTeUc4ejBlaHBac3cKPW9OT3UKLS0tLS1FTkQgUEdQIE1FU1NBR0UtLS0  
tLQo=

Hash: O5nWUEE2JycS/1sbbPYI7sVwhLNRfrvESMRVQ6M6dHg=

Version: 1.0.0

-----End Authenticated Authorship Message-----



Write a comment...



Press Enter to post.

Figure 3.10: Text Article - Full

Figure 3.9 shows the full article from Figure 3.9, with all of the metadata included.

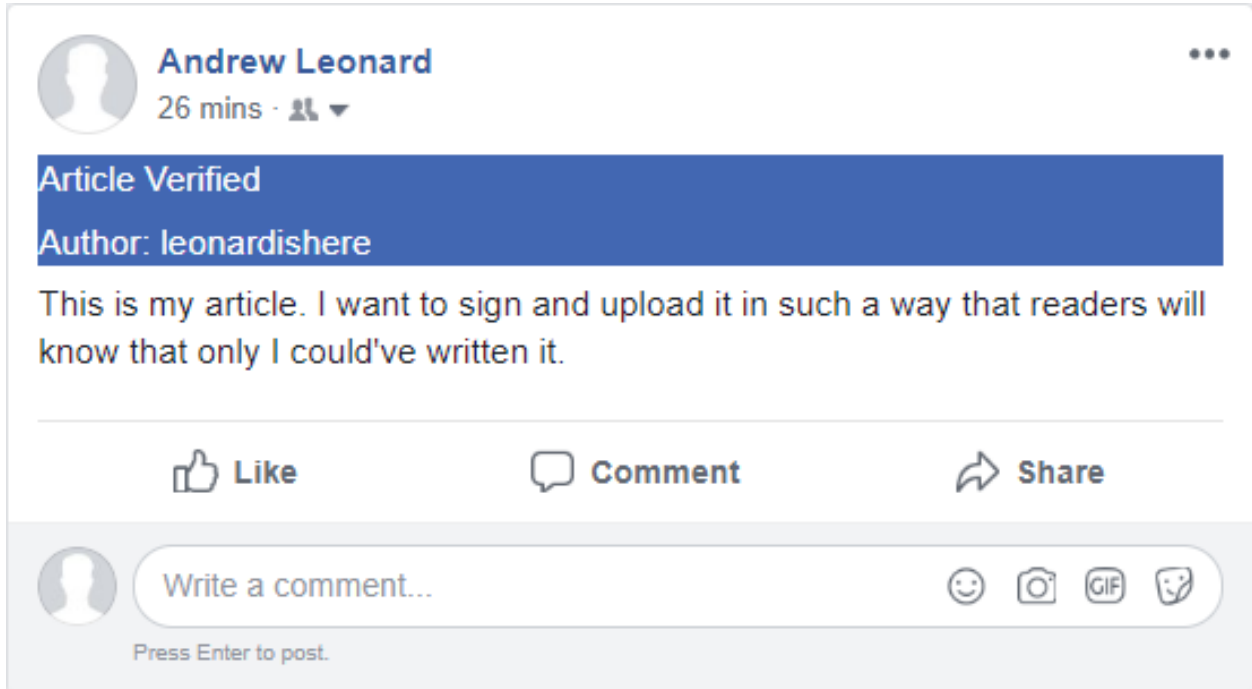


Figure 3.11: Chrome Plugin - Verified Article

Figure 3.11 shows an article verified on Facebook. If the article can be processed and properly verified by the Google Chrome plugin, all of the metadata is removed, with only the article itself remaining, and a blue notification box indicating that the article was successfully verified is added in its place, that also contains a link to the author's keybase.io page.

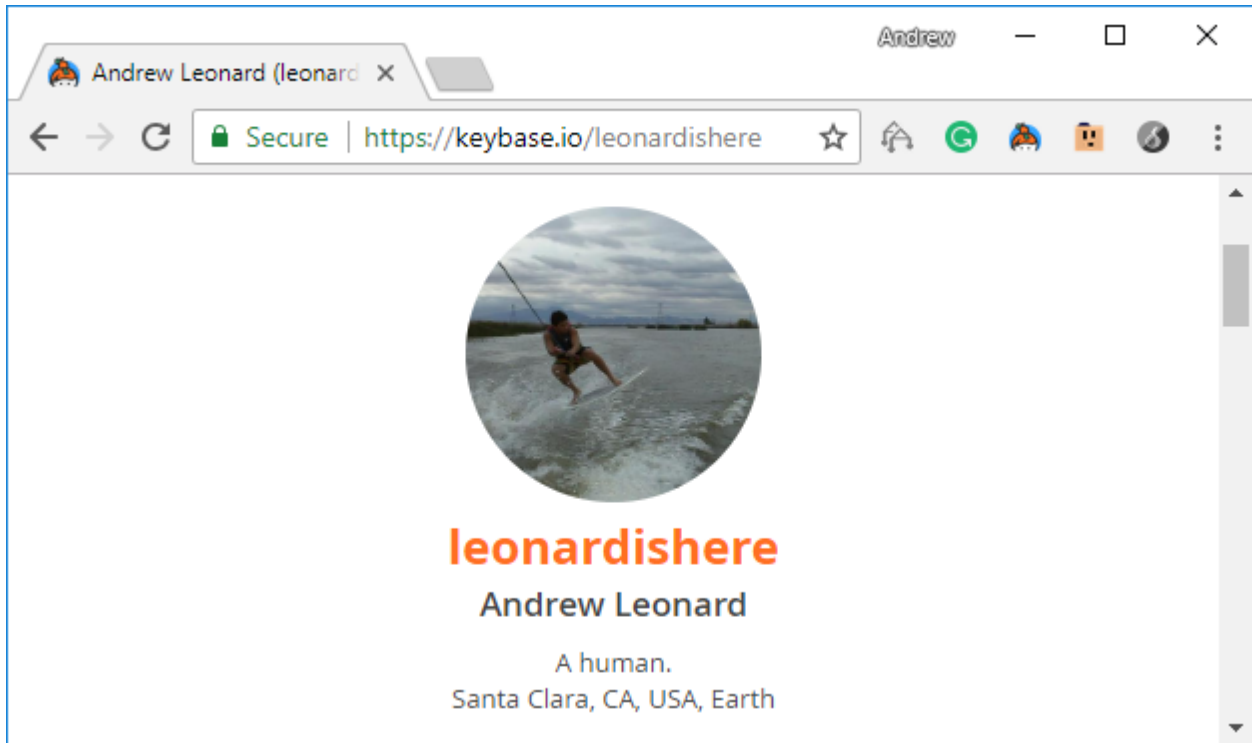


Figure 3.12: Keybase.io User Page

Figure 3.12 shows the page that results from following the link provided in the Verified Article notification. The author's keybase.io page is provided, which shows any connections the author has provided, such as Twitter, Facebook, or Github accounts.



Figure 3.13: Chrome Plugin - Failed Verification

Figure 3.13 shows the result of a failed verification due to modification of the article. The metadata is still removed, but the notification box is red to indicate that the article was not successfully verified.

## Chapter 4

# Architectural Diagram

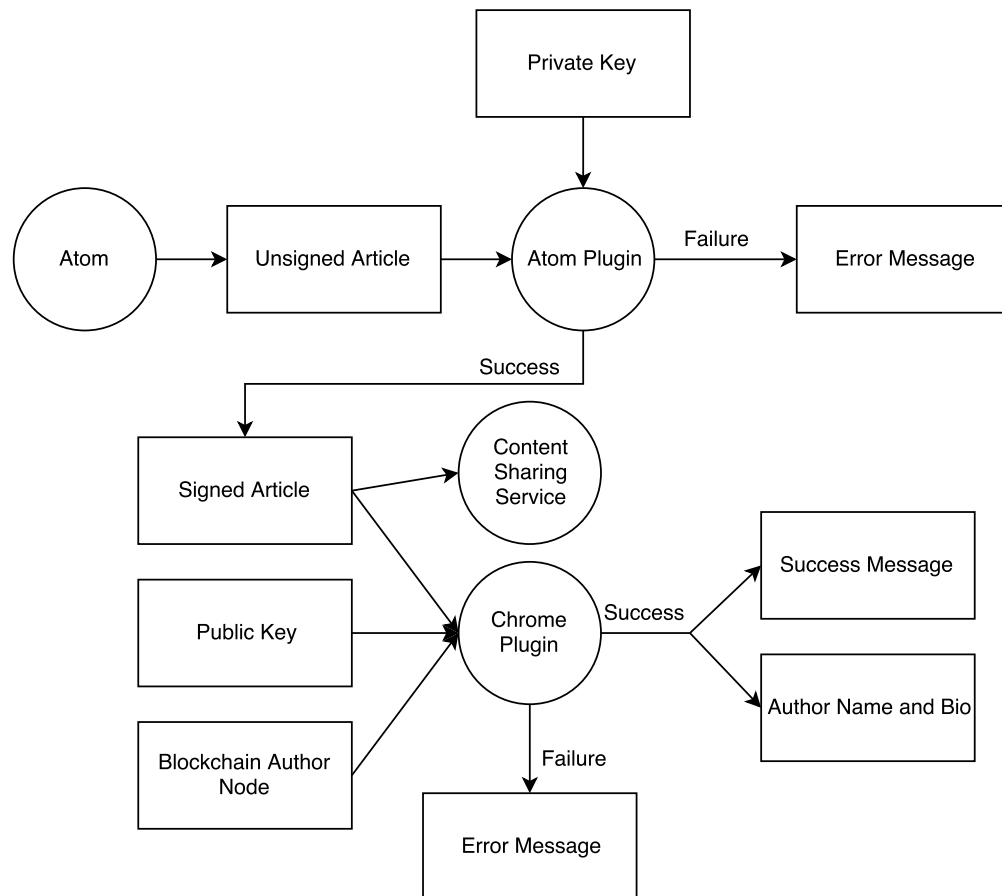


Figure 4.1: Architecture

Figure 4.1 depicts the flow of data through our system. Unsigned articles and private keys will be combined in the Atom plugin to create signed articles, unless an error occurs. The signed article can then be posted onto any online content sharing service, such as social media websites or a blog. Then, the article and public key are combined and processed using the Google Chrome extension to produce, if possible, a verified article that contains the original unsigned article's contents, along with a link to the author's keybase.io profile, which serves as their biography. If the



verification fails, the extension will notify the user with an error message.

## Chapter 5

# Technologies Used

The system uses a variety of technologies, that are described as follows.

- Atom Text Editor: Atom is an open source text editor that features easy customizability. It allows us to develop a plugin that is easy to use, and provides output in a format that is easy to utilize.
- Google Chrome Extension: Chrome is the most popular web browser, and allows for development of extensions using Javascript that provide additional functionality. Our extension will automatically parse HTML in order to recognize and verify Authenticated Authorship signed articles and modify the posts as needed.
- Keybase.io: An encryption/decryption service that will allow the user to generate their private and public key pair. Any person can create a keybase.io account and generate their own key pair, allowing them to encrypt articles through our solution.

# Chapter 6

## Design Rationale

### 6.1 Justifications for UI

- The messages displayed should be easily seen and contain lots of information. We cannot deter readers from unverified articles if they do not see the message, or if the message isn't clear.
- The author's bio will be hidden at first, but show on request. This will allow authors to have bios of arbitrary length and not clutter the screen.

### 6.2 Justifications for Technology

- Atom will be used because it is easily customizable and usable across multiple platforms<sup>[1]</sup>.
- Chrome will be used because it is the most popular web browser with 55 percent usage<sup>[2]</sup>. It is also usable across multiple platforms. It is also simple to develop in due to the documentation provided<sup>[3]</sup>.
- Keybase.IO will be used because it is a simple way for people to generate a key pair and use it to encrypt and decrypt their messages<sup>[5]</sup>. It also connects the public key to the Bitcoin blockchain using a Merkle tree<sup>[4]</sup>.

### 6.3 Justifications for Technology Not Used

- There were many text editors to choose from. We chose to not use Microsoft Word as creating a plugin would be more difficult. We also wanted to avoid stylized text and use only plain text. Other text editors, such as Vim, would also be more difficult to edit and are not always cross-platform.
- There are many web browsers that we could use. As Safari and Internet Explorer are mostly contained to particular operating systems, they were not chosen. Mozilla Firefox was another viable option, but according to StatCounter<sup>[1]</sup>, Google Chrome holds 55 percent of the market share, while Firefox holds 6 percent.
- There are other ways to store public keys and author bios. However, they use single places of storage and are easily compromised.
- We originally planned to develop a Blockchain for the encryption of the article, but found that doing so is too complicated and risky to be properly effective. Instead, we opted to use keybase.io, an existing structure that provides the functionality that we need.

- Hardware keys were planned as well but could not be properly incorporated with the private and public key pairing that we used for our application.

## Chapter 7

# Obstacles Overcome

During the project we encountered two errors which hindered our progress in the project temporarily. The two obstacles that we overcame were working with Keybase and also trying to solve problems which were not in the scope of our project.

The obstacle that we encountered with Keybase was related to them currently being in alpha while we worked on the project. We encountered the issue with them of dealing with poor APIs and documentation. They implemented their service and allow the public to use their implementation of their key management features, however, they currently have not been able to document all their code. Another problem that we encountered with them was that they were not able to provide adequate support as we were hoping to use their private key retrieval feature that they had but nobody was able to explain to us exactly what inputs we had to provide for it to work successful. To overcome the obstacle with poor APIs and documentation we spent a significant amount of time testing small chunks of their code and also our previous experiences from classes to understand what each function was suppose to do and what output it produced. To overcome the issue without having poor support we decided to implement the key storage feature that we have so that people would be able to store their key anywhere they needed to. Also we made multiple ways of signing articles.

The obstacle of trying to solve problems which were not in the scope of the project were with us the whole way through the project. This issue first arose when we were planning to implement our own blockchain for this project as it would have taking a significant amount of time and would not have been as good as the blockchains that are currently available. The way we overcame this issue was when we found Keybase and used their service as a key and author manager. The other issue arose later in the project when we thought about creating a secure key to store the author private keys on. Once again this would have required a significant amount of time and also for us to create hardware that would work securely. To overcome the obstacles of trying to solve problems that were not in the scope of the project we followed out timeline and also held weekly meeting with out advisor who helped us know if what we were working on was still in the scope of the project or starting to deviate from the project.

## Chapter 8

# Societal Issues

### 8.1 Social

We believe that society will benefit from the use of Authenticated Authorship. The main purpose of this project was to protect the identity of authors. Journalists spend their entire careers to build their reputations. They collect followers and bring value to their name through hard work and dedication to their craft, and it's unfair to allow an honest person to be attacked over words that they didn't write. We help journalists protect their identity and reputation by making it impossible for fake news to be attributed to the wrong author.

We also sought to hold authors accountable for their words. It is too easy for an author to claim that they never wrote a message; we have heard cries of "that screen shot was doctored" too often for it to be acceptable. A permanent record of the article will show that the article was written by the author and their excuses disappear. This allows for society to act on these messages properly and with valid reason.

There are also cases where anonymity on the internet is better than having identities; for example, in the case of political revolution where one would want to send a message without having it attributed to them. Because of this, we designed Authenticated Authorship to be completely voluntary. Messages can be sent without being signed by simply not using our system. Similarly, signed messages can be read without being verified.

### 8.2 Political

Political discussions, as with all discussions, are built on truth. Fake news runs rampant in politics, as highlighted during the 2016 presidential run of Donald Trump. It is important to stop fake news in this arena as it affects the lives of so many people. While our system does not determine the truth of messages, it does attribute the article to an author. This will hold journalists and politicians to their word, as well as provide a backbone to which fact checkers can operate.

## **8.3 Ethical**

The greatest ethical concern of this system pertains to the allowance of a person to be forgiven for words that they shared in the past, words that they may have realized were wrong or conveyed improperly. Our system, which would create stronger ties between the author and their messages, does not distinguish between forgivable and unforgivable messages. It merely attributes articles to authors. It is up to people to decide how to use the system and how to react to other people and their messages. This extends beyond our system and is not within our control.

## **8.4 Sustainability**

Authenticated Authorship is not directly tied to any one algorithm, key management system, or other utility. Because of this, we designed our system to be modular. This allows parts be swapped in and out as needed. Switching over to a new key management system would be as simple as changing a single call. In addition, we designed the code to be easily readable so that it can be ported to other systems. This allows for our system to be persevere through unforeseen changes.

## **8.5 Usability**

We designed Authenticated Authorship to be as usable as possible. Authors can begin signing articles with as little as a registration and a download. We created multiple possible methods of signature so that an author can choose what works for them. Readers can begin verifying articles with as little as a download. Because the Chrome extension automatically verifies articles, readers will not have to do anything besides read. Our messages are clearly visible, color coded, and easy to interpret. The extension only modifies HTML elements that pertain to the article and nothing else about the web page is changed.

## **8.6 Environmental Impact**

Our original design called for every message to be stored in the Bitcoin blockchain. Appending blocks is computationally expensive, and therefore has high energy costs. We decided to change this design so that only author blocks are stored. While it is still inefficient to add authors, it is a much more energy efficient and environmentally friendly solution. As a side note, we are not interacting with the blockchain directly; it is our key management system, keybase.io, that is tied into the blockchain.

## **8.7 Economic**

Security and cryptography are important, in demand, and on the rise. Service providers are eager to charge and rightfully so. It is tempting for us to try to make a dollar off of our users. We join keybase in their belief that every person should have a private key, and we hold that belief to a higher value than the amount we would make, even with our collective college debt. Therefore, we have decided to make our service free to use. The only expense to a user would be to purchase a private key, so we have made those optional.

Authenticated Authorship can be applied for trade purposes as well. Contracts can be signed using our system to verify the authenticity of the author, allowing for more trustful transactions. This can even be used in small online orders,

such as just a pizza from the restaurant down the street, which will allow the business to verify the user that placed the order.

## **8.8 Manufacturability**

Our system has no manufacturing costs as we have no hardware requirements. Each user will need a computer to use our system and our system requires a key management infrastructure, but none of those are manufacturing costs of our system.

## **8.9 Health and Safety**

We opted against rapid and colorful animations to reduce the chance of epileptic seizures. Besides that, no other safety concerns have been raised.

## **8.10 Lifelong learning**

In order to build this system, we had to learn many new technologies and platforms. Most of this we plan to use in our careers. However, maybe the most important lesson we learned was when to do something your self and when to use pre-existing technologies. We discovered keybase.io, which is a better key management system than what we could've built in the time allotted, platforms such as Atom that are easier for an author to use than anything we could've built, and node packages that saved us time and hassle. Additionally, Authenticated Authorship is a proof of concept and will be improved upon in the future. Knowing what we know now, a few things would've been handled differently, and we can accept that as it is a mark of lifelong learning.

## **8.11 Compassion**

We discovered honest journalists suffering when their identities were stolen, and we were compassionate to alleviate their suffering. By authenticating the authorship of articles, we hope that journalists will never have to feel that pain again.



## Chapter 9

# Conclusion

We created a system which allows authors to claim authorship of their articles. We accomplished this by making an authentication system which links a private/public key pair to an entity on the Internet. We use Keybase as a service to act as the key manager which also links the keys to the Bitcoin blockchain. We then implemented a signature feature in a text editor to demonstrate how a text editor can implement signing. We also implemented verification in the text editor so that a user can verify files that have been signed. We also created a Google Chrome Extension to provide seamless verification on the Internet and demonstrate how verification can be implemented into Internet browsers.

We learned how there are many great services on the Internet, however, they usually do not provide an easy way to use them or are just not known to the general public. We can see this with Keybase as they had something very similar to what we were planning to create, however, they were not able to make it a system which was easy to use. Even though they had this problem they provided a great service for us to use as they provided a key manager that was linked to a public ledger and also allowed us to use the keys to sign and verify through their system.

The advantages of our system are that they provide a simple way for users to use the system. Our system is also modular and upgradeable so it can continue to be relevant in the future. We also demonstrates how a text editor can be modified to use our system and also how to make a web browser compatible with the system. Some of the disadvantages of our system is that we currently only have the Chrome Extension verifying on Facebook. Another disadvantage is that we have it connected to only one public key ledger and one that is still in alpha.

Possible future work for our system is to implement a blockchain which is specific for this system. Another possible project is to create a browser extension or implementation which will work for all websites on the Internet. It is also possible to implement this solution in other text editors and also work more with Keybase to find a more simple way of retrieving the private key. There is also the possibility of creating a hardware key which will be able to store the private key from our project. Finally it is possible to link this project to other public ledgers to have a redundancy of public ledgers which will make it more secure if one of the public ledgers gets broken or goes out of commission.

## Chapter 10

# References

1. “A Hackable Text Editor for the 21st Century.” Atom, 31 May 2018, [atom.io/](https://atom.io/).
2. “Browser Market Share Worldwide.” StatCounter Global Stats, StatCounter, 8 Dec. 2017, [gs.statcounter.com/browser-market-share](https://gs.statcounter.com/browser-market-share).
3. “Develop Extensions.” Chrome: Developer, 1 June 2018, [developer.chrome.com/extensions/devguide](https://developer.chrome.com/extensions/devguide).
4. “Keybase Is Now Writing to the Bitcoin Blockchain.” Keybase, 16 June 2014, [keybase.io/docs/server\\_security/merkle\\_root\\_in\\_bitcoin\\_blockchain](https://keybase.io/docs/server_security/merkle_root_in_bitcoin_blockchain).
5. “Welcome to Keybase.” Keybase, 2018, [keybase.io/](https://keybase.io/).

# Chapter 11

## Appendix

### 11.1 Atom Install Guide

This guide will walk you through the installation process of Atom and the Authenticated Authorship Atom Plugin.

1. Download and install Atom if you haven't already. Atom's official website is <https://atom.io>. You can find the download and additional information there.
2. Open Atom. Navigate to the settings, which may depend on the operating system and the version. It should be under File → Settings or File → Preferences.
3. Click the Install button in the left menu to open the Install Packages pane.
4. Enter authenticated-authorship into the search bar and press enter. The package to install is associated with the user leonardishere. Press the blue install button to the right and wait for it to complete.

### 11.2 Keybase Account Creation Guide

This guide will walk you through creating a Keybase account and hosting a public key. This is required for authorship only.

1. Direct your web browser to <https://keybase.io>. Follow their guide to create an account and host a public key.
2. When necessary, retrieve your private key from Keybase via viewing your account at [https://keybase.io/your\\_username](https://keybase.io/your_username), then clicking the buttons as shown in the images below.

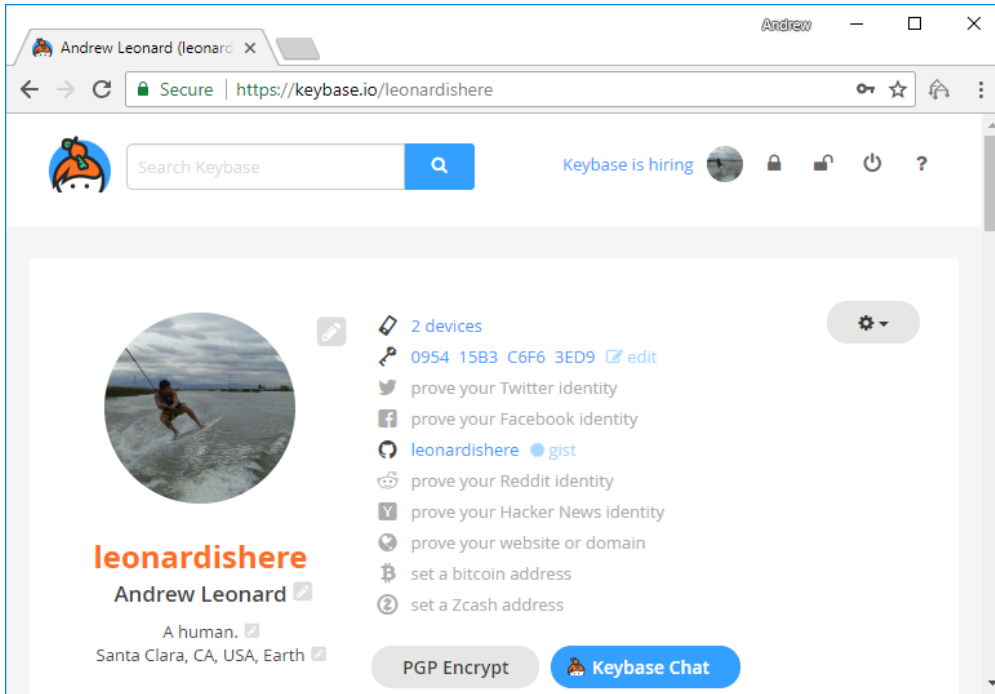


Figure 11.1: Private Key Retrieval Step 1

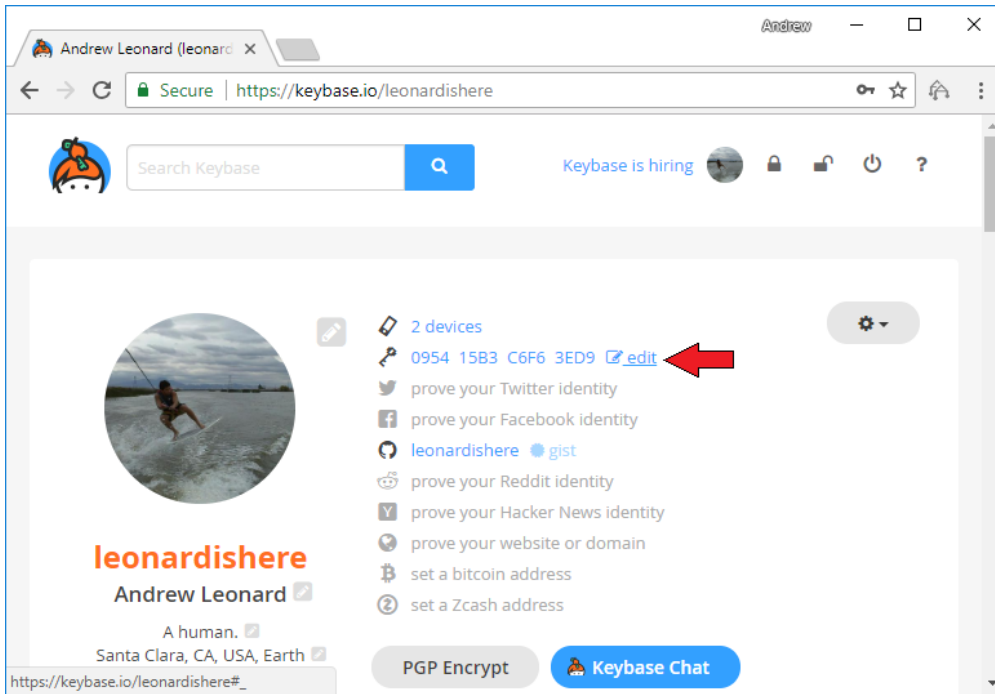


Figure 11.2: Private Key Retrieval Step 2

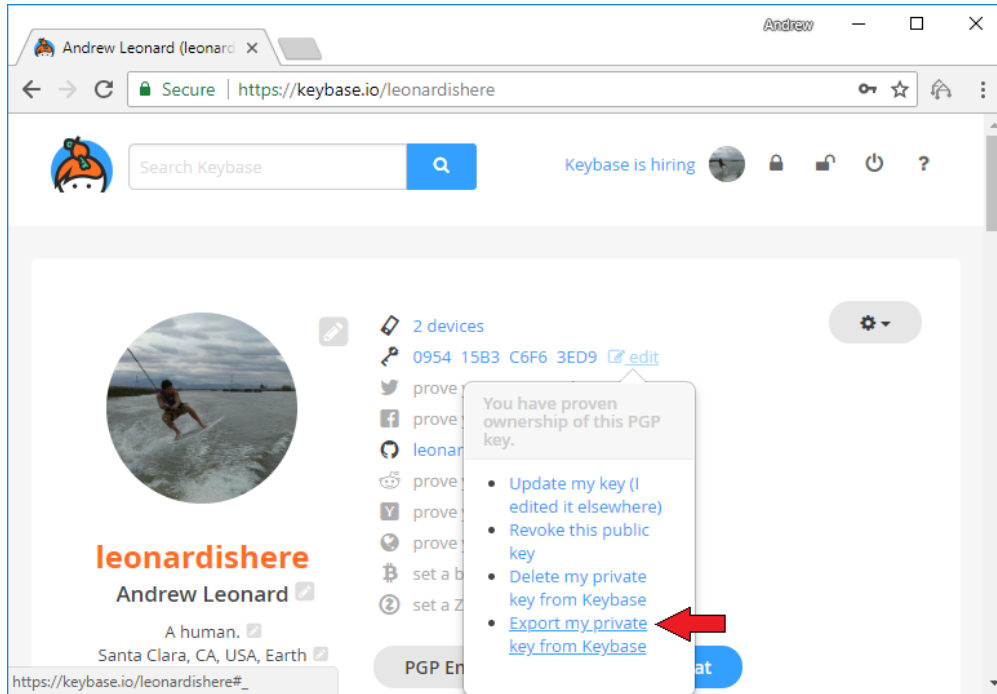


Figure 11.3: Private Key Retrieval Step 3

Follow the on screen instructions. When the private key is displayed, it can be copied and pasted into Atom.

## 11.3 Atom Use Guide

This guide will walk you through using the Atom Authenticated Authorship plugin.

### 11.3.1 Authenticating

1. Begin by writing your article in Atom. When you are finished, sign the article by using one of the following methods:
  - Ctrl-Alt-P
  - Menu Bar → Packages → Authenticated Authorship → Hardware Authenticate
  - Context Menu → Authenticated Authorship - Hardware Authenticate
2. Enter your Keybase username, password, and private key. Select the format to sign to. On success, the signed article will replace the editor text.

### 11.3.2 Store Private Key to File

1. Begin by choosing any of these methods to store your private key to a file:
  - Ctrl-Alt-I
  - Menu Bar → Packages → Authenticated Authorship → Store Private Key to File
  - Context Menu → Authenticated Authorship - Store Private Key to File
2. Enter your private key and a password to sign the private key. We recommend that this password not be identical to your Keybase password. Select the name and location of the file in the next window. Keys are stored in .AAK format, which is an abbreviation of Authenticated Authorship Key.

### 11.3.3 File Authenticating

1. Begin by writing your article in Atom. When you are finished, sign the article by using one of the following methods if you have your private key saved in a file:
  - Ctrl-Alt-L
  - Menu Bar → Packages → Authenticated Authorship → File Authenticate
  - Context Menu → Authenticated Authorship - File Authenticate
2. Enter your Keybase username and password, the key file, and the key password. On success, the signed article will replace the editor text.

### 11.3.4 Verifying

1. Do not make any edits to the file. Edits may cause authentication to fail. Verify the article by using one of the following methods:
  - Ctrl-Alt-V
  - Menu Bar → Packages → Authenticated Authorship → Verify
  - Context Menu → Authenticated Authorship - Verify

2. A status notification will appear, indicating success or failure. The editor text will also change to reflect the status.

This may be used to verify your own messages before you send them. Remember to undo any changes caused by verification before sending.

Verifying HTML articles is currently not available. You may do so by viewing them on Chrome with the Authenticated Authorship Chrome Extension installed.

### 11.3.5 Sharing

There are numerous ways to share your article. You may post it to a site via an embedded textbox, used by sites such as Twitter. Copy the article and signature in its entirety, paste it into the textbox, and post.

You may also send it as a file, like an email attachment. Save the file, upload it, and send.

To use advanced methods, such as ftp, consult your local tech expert.

## 11.4 Chrome Extension Installation Guide

This section will walk you through the installation of the Authenticated Authorship Google Chrome extension.

### 11.4.1 Enabling Chrome Extension Developer Mode

1. Direct your Google Chrome browser to `chrome://extensions/`.
2. Enable the "Developer mode" option in the top right corner of the page. This will allow you to load unpacked extensions that are not available through the Chrome Web Store.

### 11.4.2 Downloading the Authenticated Authorship Extension

1. Direct your browser to [https://github.com/leonardishere/Authenticated\\_Authorship\\_Chrome\\_Extension](https://github.com/leonardishere/Authenticated_Authorship_Chrome_Extension).
2. From the green "Clone or download" button, select the "Download ZIP" option.
3. Open the downloaded .zip file.
4. Extract the folder within the .zip file to any location you choose.

### 11.4.3 Loading the Authenticated Authorship Extension

1. Return to `chrome://extensions/`.
2. Select the "Load Unpacked" option underneath the search bar - this option is only visible if Developer Mode is enabled.
3. Navigate to the location where the extension folder was unzipped, and select the folder.
4. Click "OK" and the extension will then load into Google Chrome.

## 11.5 Chrome Extension Use Guide

This guide will walk you through using the Authenticated Authorship Google Chrome extension.

1. Direct your browser to <https://www.facebook.com/>.
2. If any posts signed by Authenticated Authorship exist on the page, the extension will automatically attempt to verify them.