

Notary

A Flask Application for Trusted Timestamping

Dario Gjorgjevski¹ Petar Tonkovikj¹

¹Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University in Skopje

October 29, 2017



Contents

- 1 Introduction
- 2 Algorithm for Trusted Timestamping
- 3 Workflow
- 4 Conclusion and Future Work



Trusted Timestamping

- Trusted timestamping is the process of securely keeping track of the creation time of a document.
- “Securely” roughly means that *no one*—not even the owner of the document—should be able to change it.
- RFC3161 specifies a scheme for trusted timestamping for the PKI
⇒ OpenSSL provides an open-source implementation of RFC3161.



Web Application

Architectural choices

- Must ensure independent timestamp generation and verification
⇒ two parts meant to be run in isolated environments.
- Both parts are built using Flask¹ and Python 3.
- Simple and bulletproof architecture built on top of the Web Server Gateway Interface (WSGI).

¹<http://flask.pocoo.org/>

Web Application

Deployment

- WSGI scripts can be deployed on the Apache HTTP Server via `mod_wsgi`.
- HTTPS and user authentication are provided by OpenSSL via `mod_ssl`.
- The certificate used for SSL is signed by our TA, Hristina.
- The certificate used for trusted timestamping is generated and self-signed by us.
- We allow only users with certificates issued by our organization to authenticate themselves to the website. See listing 1.



Web Application

Authentication

```
# Access Control:
# With SSLRequire you can do per-directory access control based
# on arbitrary complex boolean expressions containing server
# variable checks and other lookup directives. The syntax is a
# mixture between C and Perl. See the mod_ssl documentation
# for more details.
<Location>
SSLRequire %{SSL_CLIENT_I_DN_O} in \
    {"Faculty of Computer Science and Engineering", \
     "Ss. Cyril and Methodius University", \
     "FCSE", "UKIM"}
</Location>
```

Listing 1: OpenSSL Access Control



Contents

- 1 Introduction
- 2 Algorithm for Trusted Timestamping**
- 3 Workflow
- 4 Conclusion and Future Work



Generating a Trusted Timestamp

Informally

Generating a trusted timestamp is straightforward and consists of:

- ① Hashing the data that needs to be timestamped;
- ② Concatenating the hash with the timestamp;
- ③ Hashing the concatenation of the two;
- ④ Digitally signing the hash with the private key of the TSA.



Generating a Trusted Timestamp

Formally

A trusted timestamp contains **data**, a **timestamp**, a hash H , and a signature S . The latter two are defined as:

$$\begin{aligned} H &:= H(H(\text{data})\|\text{timestamp}), \text{ and} \\ S &:= \text{Sig}_{\text{TSA}}(H), \end{aligned} \tag{1}$$

where $\text{Sig}_{\text{pk}}(\cdot)$ is a digital signature under the private key pk , $H(\cdot)$ is a cryptographic hash function such as SHA-256 or SHA-512, and “ $\|$ ” denotes concatenation.



Verifying a Trusted Timestamp

- ① Hash data and tentative timestamp as previously.
- ② Verify the TSA's signature on the provided hash.
- ③ Compare the computed hash and the provided hash.

The two hashes being identical is a strong guarantee of the timestamp being correct (i.e., issued by the TSA) and unaltered.



Contents

- 1 Introduction
- 2 Algorithm for Trusted Timestamping
- 3 Workflow**
- 4 Conclusion and Future Work



Supplying the Authentication Certificate

All major web browsers are able to supply PKI certificates for authentication to the website. An example doing this on our website is found on figure 1.

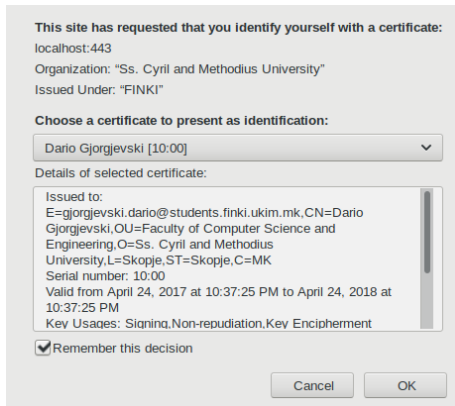


Figure 1: Mozilla Firefox Prompting for Certificate



Verifying the SSL Connection

The screenshot displays a web browser window with the address bar showing `https://localhost`. The main content area is titled "Notary" and includes a "Browse" button with a "No file selected." message and an "Upload" button. Below this is a table listing files:

Filename	Size	Last modified	Timestamp	Delete
Photos				
Abstract.txt	1.3 kB	18 hours ago	Timestamp	Delete

Below the table, it states "1 file(s) and 1 folder(s)."

Overlaid on the bottom left is the "Certificate Viewer: 'Scotman'" window. It shows the following details:

- General:** Could not verify this certificate because the issuer is unknown.
- Issued To:** Common Name (CN) **Scotman**, Organization (O) **Ss. Cyril and Methodius University**, Organizational Unit (OU) **Faculty of Computer Science and Engineering**, Serial Number **10:00**.
- Issued By:** Common Name (CN) **BNKS CA**, Organization (O) **FINKI**, Organizational Unit (OU) **BNKS Certificate Authority**.
- Period of Validity:** Begins On **May 11, 2017**, Expires On **May 11, 2018**.
- Fingerprints:** SHA-256 Fingerprint **CE:6A:31:A6:32:28:75:E2:E5:06:41:92:82:FA:0E:DE:EC:F8:F9:9A:60:EF:03:EA:90:51:5F:50:3C:05:CC:FB**, SHA1 Fingerprint **F7:9E:D0:98:0C:80:7C:4C:A2:E3:5E:2D:4B:75:1A:CA:31:94:30:A8**.

Overlaid on the bottom right is the "Page Info - https://localhost" window. It shows:

- Website Identity:** Website: **localhost**, Owner: **This website does not supply ownership information.**, Verified by: **FINKI**, Expires on: **May 11, 2018**. (Buttons: [View Certificate](#))
- Privacy & History:** Have I visited this website prior to today? **Yes, 185 times**; Is this website storing information (cookies) on my computer? **Yes**; Have I saved any passwords for this website? **No**. (Buttons: [View Cookies](#), [View Saved Passwords](#))
- Technical Details:** Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bit keys, TLS 1.2). The page you are viewing was encrypted before being transmitted over the Internet. Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network. (Buttons: [Help](#))

Figure 2: SSL Certificate Information

Uploading Documents for Trusted Timestamping

- Uploading is as simple as pressing the **Upload** button.
- Once on the server, document can be either:
 - ▶ Timestamped; or
 - ▶ Deleted.
- Pressing the **Timestamp** button sends to the user a zip file with a:
 - ▶ Timestamp query (TSQ) used to timestamp the file; and a
 - ▶ Timestamp response (TSR) containing the trusted timestamp.



Uploading the TSQ and TSR for Verification

The TSQ and TSR files downloaded from the file server can be uploaded to the second web application for verification. Once uploaded, they are passed as input to the `ts -verify` OpenSSL command.



The screenshot shows a web application interface with a light blue header bar containing the text "Upload new File". Below the header, there are two rows of input fields. The first row is labeled "Timestamp query:" and contains a "Browse..." button and a text field with the value "Abstract.txt.tsq". The second row is labeled "Timestamp response:" and contains a "Browse..." button and a text field with the value "Abstract.txt.tsr". Below these fields is an "Upload" button.

Figure 3: Uploading the Required Files

Inspecting the Verification Results

Once a user has uploaded the TSQ and TSR files, he/she will be redirected to a page containing the results of the verification.

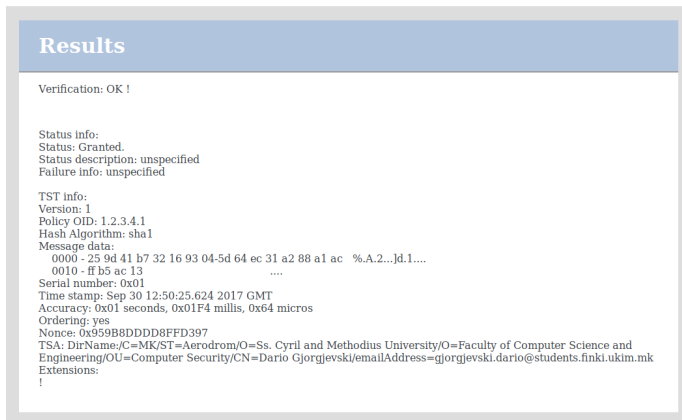


Figure 4: Output of `ts -verify`



Pros and Cons of Having Two Separate Applications

Pros

- Users do not need to use OpenSSL at all \Rightarrow more user-friendliness and less error-proneness.
- Separate entities for timestamping and verification \Rightarrow more trustworthiness since the compromise of one entity can be “detected” by the other.

Cons

- Deployment: it is harder to deploy and keep two applications up-to-date.
- Convenience: users need to use two different applications for a single functionality.



Contents

- 1 Introduction
- 2 Algorithm for Trusted Timestamping
- 3 Workflow
- 4 Conclusion and Future Work



Conclusion

- We developed a simple trusted timestamping server.
- Suitable for deployment in *closed* environments: closed in a sense that users can authenticate themselves via certificates.
- All components utilized are open-source, simple, and have been used and studied extensively.



Future Work

- Store files on a database for easier fault-tolerance and backups. At present, files are stored on the file system as-is.
- Functionality to timestamp files the moment they are uploaded rather than upon request.
- Alternative ways of authentications (user accounts).
- Private files.
- Same certificate for the TSA and the SSL connection.

