Sign-Verify

By: Rivka Nadell and Sara Teigman

Final project, semester 1, year 2021.

Computer Science Department at Jerusalem College of Technology

Instructor: Barak Einav

Table of Contents

[Introduction 1](#_Toc61982420)

[Project Purpose 1](#_Toc61982421)

[Why Choose Us? 2](#_Toc61982422)

[Design & Architecture 2](#_Toc61982423)

[Project Flow 2](#_Toc61982424)

[Flow Explanation 3](#_Toc61982425)

[How To Use Our Project 3](#_Toc61982426)

[Security analysis 3](#_Toc61982427)

[Mode of Work 3](#_Toc61982428)

[Challenges We Faced 3](#_Toc61982429)

[The Future of our Project 4](#_Toc61982430)

Sign and verify + description

# Introduction

The name of our project is Sign-Verify. It’s a google chrome extension for Gmail that uses asymmetrical keys to confirm the identity of the sender of a signed email. The sender can use a private key to sign the contents of an email and send it to the receiver. The receiver in turn can use the public key of the sender to confirm that it is indeed an authentic email received from the correct source.

# Project Purpose

The purpose of our project is to allow the users to authenticate the source of their emails. This allows for message integrity, especially when receiving sensitive data.

## Why Choose Us?

Why is better – signature in DAL so it’s for sure safe. Lighter than others with more features.

Worse – graphic interface. Limited features (nothing else). Makes email ugly. Only one account per computer with a private key saved.

Our email authentication project is better than other existing projects of its kind in many ways. Among them, the private key which allows email signing is stored in the flash storage of the DALapplet, which is a well protected storage area. Other existing solutions store the private key in the regular software or operating system in use.

Another reason why our solution trumps existing solutions is that since we only have the one feature of signing and verifying emails, it’s a very light extension to download and doesn’t come with a bunch of unnecessary files. The limited features in our project make it light, but it can also be a downside. Some users may want an extension that comes with a few other security features that ours doesn’t yet offer.

# Design & Architecture

## Project Flow

Extension injected

Email client

Extension backround

Extension content



server



DAL applet

C# - host app

## Flow Explanation

# How To Use Our Project

How to use project: install extension. run server. Run c#. open email. And click on compose button or respond to an email and click on the relevant button – sign or verify. And watch the magic unfold

# Security analysis

– TBD

Work: mostly together. Some backend DAL sign and C# verify and server – Rivka. Frontend messaging, verify and lots of research and time and effort– Sori.

Challenges: communicating between parts. Buttons and verify. Access to info.

# Mode of Work

While working on this project we found it easier to work together most of the time. We would share the screen from one of our computers and research and code that way. We did end up working on own towards the end of the project with the known parts of it.

Some of the backend with the DAL, C# and server was done by Rivka,

and the frontend messaging between files was done by Sara.

## Challenges We Faced

During each step of our project journey we encountered many obstacles and challenges. Starting with the front-end, we had little to no background on the JavaScript language as well as all the add-ons that go with it. We also didn’t really know anything about creating our own chrome extension. Most of these challenges we were able to overcome due to hours upon hours of research and google searches. We tried a lot of different ways of doing the above until we finally got a working project and then built up from there. For the back-end part of the project, we did know what we were doing because it was using the material we learned about in class and we had to alter it to our requirements. It did take a lot of debugging and fixing things up, but ended up working in the way we wanted. The really big challenge for us was the connection between the chrome extension and our C# host app. We tried a lot of ideas that we found in our research but nothing seemed to work. After almost giving up multiple times we finally tried to use a local server to connect the two parts of the project. Adding the server as a component in our project was also a lot of new research and learning how to use it with two programs connecting to it, but that was finally the right idea to give us the connectivity that we needed in order for our project to work as it should.

# The Future of our Project

Future work:

encrypt and decrypt email content.

Allow extension to work from multiple accounts per computer.

Make nicer graphic interface.

Make possible to run without running server and c# app first individually.

[Save public key in file to verify that it’s the same one each time for each email so that no hacker uses same email address and different private and public key.

Make verifying possible not only from compose email right after signed.]

.

Why? What? How?