Exposé – Specification for a distributed secret management system with sharing capabilities

Paul Bienkowski

2bienkow@informatik.uni-hamburg.de

June 10, 2016

1 Motivation

The most common way for humans to authenticate against digital systems nowadays is the use of passwords, the core reason for which is its obvious simplicity. Compared to other methods such as cryptographic keys, devices, or biometric feature detection, it is very simple to create, use and verify passwords, both for the system as the user.

Furthermore, passwords can simply be shared among people. While this practice is generally discouraged due to its security implications, it still finds widespread use within teams. Instead of creating separate accounts for each team member, and updating them whenever someone joins or leaves the team, a group password is generated and everyone in the team is given access via this password. This reduces the management overhead for the team leaders, and it becomes very important when different organizations work together using resources made available through one of the organization's digital services.

Given the ever-increasing computational capacity of computers, one should not utilize simple passwords that are easy to remember for authentication. Instead, it is considered best practice to generate long random passwords from a large character set, and not reuse them for different systems. This ensures that the password can neither be guessed nor brute-forced in reasonable amounts of time. In this manner, many companies enforce a strict password guideline among their employees. Because it is near impossible to remember many of those passwords, password managers have come into existence, to store all of a user's passwords using encryption technologies to keep them safe. Then, only a single *master password* is required to unlock all personal passwords stored inside.

We consider a team of employees, each utilizing a password manager (as enforced by company policy) to manage their personal passwords. Sharing team-wide passwords via (possibly unencrypted) email or other text-based media is always a security risk, and sharing them orally (via phone or in person) is tedious and may result in errorneous transmission. Instead, the company seeks to use a common password manager that is capable of sharing passwords with other people.

There are plentiful solutions for this, where one can transmit a password secretly to a single contact, or group of contacts. However, these existing solutions have one problem in common: they all are proprietary (often paid) solutions, hosted by their creators *„in the cloud"*, that is, on a publicly accessibly system. While this makes it very simple to set up a team to use these solutions, and generally works pretty well for everyone, it also involves a security risk: highly confidential data (secrets used for authentication) are stored on foreign servers, made accessible via the internet, outside the control of the data owners.

Furthermore, these servers are a high profile target for attackers, since they store secrets for many of their clients. The popular password management site LastPass has just recently (2015) been target of an attack, where login hashes have been exposed [2]. Like that, a single successful attack might allow access to many more protected resources – in above notice LastPass claims to serve „millions of users".

In my thesis, I plan to design a system that does not involve this security risk: a distributed secret management system with sharing capabilities.

2 Goal

The goal of this work is to specify a protocol for a distributed system that resolves the introduced problems. A system following that protocol shall be able to synchronize secrets in a secure manner, such that multiple people can concurrently access and modify those secrets.

This system shall exhibit these main features:

- **Security**: Since the system is handling secrets, the solution must provide secure storage and transmission. This is the core priority of the protocol.
- **Distributed storage**: The storage shall be distributed, i.e. accessible from different devices, and synchronizable across those devices.
- **Sharing**: It must be possible to share secrets with other people, in a manner that ensures those people can access the secret, but nobody else.
- **Conflict resolution**: The system must provide conflict resolution routines, that is, specify how to handle the event of different changes to the same secret to reach a consistent state.

3 Results

As a result of this work I want to produce a finished specification that can be used to implement a system for the described goal.

In order to develop a working specification, I shall first design a draft, implement a prototype according to that draft, use the prototype to evaluate the draft and then revise the draft to a final specification using the learnings extracted from the evaluation.

4 State of the Art

To my knowledge there is currently no system that exhibits the desired features. However, there are lots of tools that provide a partial featureset. Here is a small overview, the best of these tools

1. Introduction
   - 1.1. Motivation
   - 1.2. Goals
2. State of the Art
   - 2.1. Specification formats
   - 2.2. Password managers
   - 2.3. Encryption tools
   - 2.4. Distributed data storage and synchronization
   - 2.5. Conflict resolution
3. Derivation of the Specification
   - 3.1. Secret format
   - 3.2. Encryption and sharing
   - 3.3. Synchronization and transmission
   - 3.4. Conflict resolution, un-sharing
4. Discussion / Evaluation
   - 4.1. Strengths
   - 4.2. Problems
   - 4.3. Adjustments made to the specification
5. Conclusion
6. References

Table 1: Preliminary Table of Contents

combined may be a good starting point for the newly designed system:

OpenPGP[1] is a widespread standard for personal message encryption and signing. I have found implementations of password managers using OpenPGP, such as pass[1]. Another notable password manager is KeePass[2], with a well defined file format, and a synchronization plugin (though this is single-user only).

There are many different code version control systems, such as git [3] and Apache Subversion[4] that may provide mechanisms for both distributed storage (and sharing) and conflict resolution.

References

[1] OpenPGP Message Format. `https://tools.ietf.org/html/rfc4880`, 2007.

[2] LastPass Security Notice. `https://blog.lastpass.com/2015/06/lastpass-security-notice.html/`, 2015.

[1] `http://passwordstore.org`

[2] `http://keepass.info/`

[3] `https://git-scm.com/`

[4] `https://subversion.apache.org/`

Week	Activity	Writing goal
1	Literature Research, State of the Art	Introduction & Goals
2		
3	Research specification formats	State of the Art
4	Define requirements	
5		
6	Experiment with software architecture	
7		
8	Prototype implementation & learning	Specification & Prototype
9		
10	Convert learnings to specification	
11		
12	Review & finalize specification	Evaluation
13		
14	Collect findings and arguments into thesis	Discussion
15		
16		Review & finalize thesis
17		
18		
19	*Buffer*	
20		
21	Submission of thesis	

Table 2: Estimated Timetable