A protocol for a distributed secret management system with sharing capabilities

Exposé for a Bachelor of Science thesis

Paul Bienkowski

2bienkow@informatik.uni-hamburg.de

July 4, 2016

1 Motivation

The most common way for humans to authenticate to 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 understand* how passwords work and how to use them, at least on a basic, rather insecure level.

Furthermore, passwords can be shared with other people. While this practice is generally discouraged for security reasons, 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.

A problem arises when one aims at using passwords in a secure manner. 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 passwords them for different systems. This ensures that the password can neither be guessed nor brute-forced in reasonable amounts of time. To achieve this, many companies enforce a strict password guideline among their employees. Because it is near impossible to remember a large number of those high-quality passwords, password managers have come into existence, which store all of a user's passwords using encryption technologies. Then, only a single *master password* is required to unlock all personal passwords stored inside, which the user must remember (and keep safe).

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 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 numerous solutions for this, which allow transmission of a password secretly to a single contact, or group of contacts. However, these existing solutions have one problem in common: they are all proprietary and 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, the 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 an interesting target for attackers, since they store many secrets for large companies. The popular password management site LastPass has just recently (in 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 – 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 solves the introduced problems. A system following this 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 specification for a protocol that can be used to implement a system for the described goal.

In order to develop a working protocol, 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 version using the learnings extracted from the evaluation.

Table 1: Preliminary Table of Contents

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 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 provide mechanisms for both distributed storage (and sharing) and conflict resolution. I expect to be able to adapt one of these mechanisms to the problem at hand.

[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 protocol specification formats	State of the Art
4	Define requirements	
5		
6	Experiment with software architecture	
7		Protocol specification
8		
9	Prototype implementation	
10		
11		Evaluation
12		
13		
14	Improvements	Discussion
15		
16		
17	*Buffer*	Review & finalize thesis
18		
19		
20		
21	Submission of thesis	

Table 2: Estimated Timetable

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.