File Comparator based on Zero-Knowledge Proof

Raziq R. Ramli
CS Department, Purdue University

November 2021

1. Problem Description

Alice and Bob are subcontractors for the same company. They each hold a 4GB file of all the company clients' passwords and they are supposed to use them to develop apps. To make sure the apps are consistent they need to ensure that the password files they hold are identical. But ... they do not trust each-other!

Your goal is to implement a protocol which will allow them to check that the files are identical without any of the parties revealing to the other party the contents of his file.

2. Code Spec

Language used:

- Python

Libraries used:

- socket - for client-server implementation
- hashlib - for hashing messages
- secrets - for generating large random keys securely
- nacl - for signing/verifying messages

External tool used:

- openssl dhparam - for generating large prime multiplicative groups and their generators

3. Approach

We reduced the problem to the Socialist Millionaire Problem, where two millionaires want to check if they have equal wealth without having to disclose their amount of wealth.

Then, we incorporated an existing solution to the problem called Socialist Millionaire Protocol (SMP) as a routine in our protocol. The general outline for our protocol is as follows.

Initial Routine

- Alice computes m_a = SHA-3(Alice's file)
- Bob computes m_b = SHA-3(Bob's file)
- Alice and Bob agrees on a cyclic group G of prime order q (2048-bit), and a generator g_1 (this information is public)

SMP - Key Exchange 1

- Alice generates random numbers a_2 and a_3 s.t. $q \nmid a_2, a_3$
- Bob generates random numbers b_2 and b_3 s.t. $q \nmid b_2, b_3$
- Alice sends $g_1^{a_2}$ and $g_1^{a_3}$
- Bob sends $g_1^{b_2}$ and $g_1^{b_3}$
- Alice and Bob compute $g_2 = (g_1^{a_2})^{b_2}$ and $g_3 = (g_1^{a_3})^{b_3}$

SMP - Key Exchange 2

- Alice generates a random number r
- Bob generates a random number s
- Alice sends $P_a = g_3^r$ and $Q_a = g_1^r g_2^{m_a}$
- Bob sends $P_b = g_3^s$ and $Q_b = g_1^s g_2^{m_b}$

SMP - Key Exchange 3

- Alice sends $R_a = (Q_a/Q_b)^{a_3} = g_1^{(r-s)a_3} \cdot g_2^{(m_a-m_b)a_3}$
- Bob sends $R_b = (Q_a/Q_b)^{b_3} = g_1^{(r-s)b_3} \cdot g_2^{(m_a-m_b)b_3}$
- Alice and Bob compute $R_{ab} = ((Q_a/Q_b)^{a_3})^{b_3} = g_1^{(r-s)a_3 b_3} \cdot g_2^{(m_a-m_b)a_3 b_3}$

SMP - Key Comparison

- Alice's and Bob's files are equal if $R_{ab} == P_a/P_b$

Proof of correctness :

$$
\begin{aligned}
R_{ab} &= P_a/P_b \\
g_1^{(r-s)a_3b_3} \cdot g_2^{(m_a-m_b)a_3b_3} &= g_1^{(r-s)a_3b_3} \\
\cancel{g_1^{(r-s)a_3b_3}} \cdot g_1^{(m_a-m_b)a_2b_2a_3b_3} &= \cancel{g_1^{(r-s)a_3b_3}} \\
g_1^{(m_a-m_b)a_2b_2a_3b_3} &= 1 \\
(m_a-m_b)a_2b_2a_3b_3 &= kq && \text{for some } k\epsilon\mathbb{Z}\text{, since } g_1 \text{ has order } q \\
(m_a-m_b)a_2b_2a_3b_3 &= 0 && \text{since } q \text{ is prime \& } q \nmid (m_a-m_b), a_2, b_2, a_3, b_3 \\
m_a - m_b &= 0 && \text{since } a_2, b_2, a_3, b_3 \neq 0 \\
m_a &= m_b
\end{aligned}
$$

$m_a = m_b$ implies SHA-3(Alice's file) = SHA-3(Bob's file).

From the collision-resistance property of SHA-3, we can conclude with a high probability that Alice and Bob share the same file.

4. Security Goals

For simplicity, we will call our adversary Eve, and we may only refer to Alice when writing from the perspective of honest parties. However, no loss of generality is implied.

Goal 1 *Even if Alice knows the full content of Bob's file (say, from past protocol interactions), Alice will not be able to learn anything about the file that Bob sends over this protocol.*

Exception :

Alice may learn about Bob's sent file if the protocol indicated that their files are equal.

Approach :

Socialist Millionaire Protocol, as described in Section 3.

Proof :

If Bob's file is not equal to Alice's file, we will prove that Alice will not be able to learn anything about the file that Bob has sent over this protocol.

Bob has only sent his file m_b, to Alice in the form of

$$Q_b = g_1^s g_2^{m_b}$$

and

$$R_b = g_1^{(r-s)b_3} \cdot g_2^{(m_a-m_b)b_3}$$

From hardness of Discrete Log problem, these two message do not expose the content of m_b.

However, suppose Alice knows the content of Bob's old file, m_b', the hardness of Discrete Log problem does not prevent Alice from trying to compute

$$Q_b' = g_1^s g_2^{m_b'}$$

$$R_b' = g_1^{(r-s)b_3} \cdot g_2^{(m_a - m_b')b_3}$$

If the computed $Q_b' = Q_b$ or $R_b' = R_b'$, then Alice can be certain that $m_b = m_b'$.

However, computing Q_b' or R_b' requires Alice to know Bob's secret key s, which is not exposed anywhere throughout the protocol. Thus, Q_b' and R_b' are not computable by Alice, and Alice cannot learn about the file that Bob has sent over this protocol, even if she has the full copy of Bob's file on her machine.

Goal 2 *Even if Eve has access to the messages previously sent by Alice, Eve should not be able to impersonate as Alice to Bob undetected.*

Approach :

To prevent Eve from sending forged messages to Bob undetected, Alice will always sign the message that she sends with her private key, and Bob will always verify the signature that he receives with Alice's public key. The exact public-key digital signature system being used is Ed25519.

To prevent Eve from randomly sending Alice's old messages to Bob undetected, Alice will always append the message that she's sending with the message that she recently receives from Bob. Bob, in turn, will always verify that the message he receives contains the message that he recently sent as a suffix.

Supposed Bob detects any forgery or invalid suffix, then he will abort the protocol.

Proof :

From the security of Ed25519 signature system, it is infeasible for Eve to forge a valid $(message, signature)$ pair.

From the randomness of the messages in Socialist Millinoaire's Protocol (and therefore the randomness of the suffix in Alice's messages) and the size of the message space in Socialist Millionaire's Protocol, it is infeasible for Eve to find a valid message that has been signed by Alice.

Goal 3 *At the end of the protocol, even if eavesdropper Eve knows the full content of Alice's file, she should not be able to tell if Alice and Bob have the same files or not.*

Approach :

Socialist Millionaire Protocol, as described in *Section* 3

Proof :

Eve may only learn that Alice's and Bob's files are equal if she can compute

$$P_a/P_b$$

and

$$R_{ab} = (Q_a/Q_b)^{a_3 b_3}$$

However, to compute $R_a b$, Eve would need to obtain Alice's secret key a_3 or Bob's secret key b_3, which were never exposed anywhere in the protocol. So, it is not possible for Eve to learn the equality of Alice's and Bob's files through this protocol, even if she has the full content of Alice's file.

Consequence :

Even if Alice's password file was somehow leaked to Eve, this protocol will not allow Eve to test if she also knows the full content of Bob's password file.

5. Assumptions

For the protocol implementation, is assumed that:

- Alice and Bob knows each others' public keys in advanced
- Alice and Bob has fixed a public prime number p and generator g_1 for the Socialist Millionaire Protocol in advanced
- Alice and Bob may only perform passive attacks based on what is on their view

5.Note

For simplicity, the implemented server only allows 2 connections at a time. This may allow an adversary to monopolize the server connections and deny services to Alice or Bob. However, since this project is only concerned with data secrecy and integrity, this type of attack is not covered in the implementation.