Secure Comparator based on Zero-Knowledge Proof

Raziq R. Ramli
CS Department, Purdue University

December 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. Solution

We can reduce 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.

Given this, we can implement a protocol that uses the Socialist Millionaire Protocol (SMP) which is a solution to the socialist millionaire problem, to achieve this project's goal. The general outline for the implemented protocol is as follows:

Initial Routine

- Alice computes m_a = SHA3-512(Alice's file)
- Bob computes m_b = SHA3-512(Bob's file)
- Alice and Bob agrees on a cyclic group G of prime order q (2047-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 $R_{ab} = ((Q_a/Q_b)^{a_3})^{b_3} = g_1^{(r-s)a_3b_3} \cdot g_2^{(m_a-m_b)a_3b_3}$

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 SHA3-512(Alice's file) = SHA3-512(Bob's file).

From the collision-resistance property of SHA3, we can conclude that Alice and Bob share the same file.

3. Security Goals

Without loss of generality, this report will call the adversary Eve, and will only refer to Alice when writing from the perspective of honest parties.

In addition, note that the security goals are designed with the assumption that honest parties may only perform passive attacks to each other.

Goal 1 *The protocol should prevent Alice from learning the content of Bob's file, even if Alice possesses a full copy of the file (e.g., from past successful comparisons).* [1]

Approach :

Socialist Millionaire Protocol, as described in Section 2.

Proof :

The Socialist Millionaire Protocol requires Bob to send his hashed file, m_b, in the form of

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

From the hardness of the discrete logarithm problem, these two message do not expose the value of m_b.

However, suppose Alice knows the content of Bob's past file, m_b' (from past successful comparison) the hardness of the discrete logarithm problem does not prevent Alice from trying to compute

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

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

Assuming Alice can compute Q_b' or R_b', if $Q_b' = Q_b$ or $R_b' = R_b$, then Alice can be certain that $m_b = m_b'$.

However, computing Q_b' and R_b' requires Alice to know Bob's secret key s, which the protocol protects using randomness and the hardness of the discrete logarithm problem. So, Q_b' and R_b' are not computable by Alice

Thus, Alice cannot use the protocol to learn the content of Bob's file, even if she posseses a full copy of it.

Goal 2 *The protocol should prevent Eve from impersonating as Alice undetected, even if Eve has access to the messages previously sent by Alice.*

[1] Alice may learn the content of Bob's file if the protocol indicated their files are equal.

Approach :

The protocol incorporates Ed25519 public-key digital signature system to authenticate the channel between Alice and Bob.

In other words, the protocol requires Alice to sign the message that she sends with her private key, and the protocol requires Bob to verify the $(message, signature)$ pair that he receives with Alice's public key.

To prevent Eve from randomly sending Alice's past $(message, signature)$ pairs to Bob undetected, Alice will append the message that she is sending with the message that she recently receives from Bob. Bob, in turn, will verify that he receives from Alice is appended with his recently sent message.

If any of the verfication fails, then Bob 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 Alice.

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 Protocol, it is infeasible for Eve to find a valid message that has been signed by Alice.

Goal 3 *The protocol should prevent eavesdropper Eve from learning if Alice and Bob have the same file or not.*

Approach :

Socialist Millionaire Protocol, as described in *Section* 2

Proof :

The Socialist Millionaire Protocol allows any party to learn the equality of Alice's and Bob's files if they can compute both

$$P_a/P_b$$

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

However, to compute R_{ab}, Eve needs to obtain Alice's secret key a_3 or Bob's secret key b_3, which the protocol protects using randomness and the hardness of the discrete logarithm problem. So, R_{ab} is not computable by Eve.

Thus, it is not possible for Eve to learn the equality of Alice's and Bob's files through this protocol.

4. Implementation

4.1 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

4.2. Assumptions

For the protocol implementation, is assumed that:

- Alice and Bob knows each others' public keys in advanced
- Alice and Bob fixes a cyclic group G of prime number q and generator g_1 for the Socialist Millionaire Protocol in advanced

4.3. 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.