Team Members :

- Digvijay Rajesh Pawar (dpawar@purdue.edu)
- Joshua Chukwuemeka Omoke-Enyi (jomokeen@purdue.edu)
- Tae Yoon Kim (kim2942@purdue.edu)

Team Name : BlockchainForever

Coding Language : Python

Github Repository : https://github.com/thinkty/fa23-cs355-group-project

Version : 3e8e787c919cdadd5a1a611f4beec2bddc0a21aa

Modules Used: cryptography, socketserver, argparse, hashlib

Protocol Assumptions:

- Bob is able to obtain Alice's public key.
- Bob and Alice do not have access to other segments that they don't have.

Protocol Specification

- Security Goal : If the RSA assumption holds, and H is an irreversible hash function that is modeled as a random function mapping onto $\mathbb{Z}^*N$, then RSA-FDH is secure (lecture 14 slide 29) s.t. the adversary should not be able to forge a valid signature on any message not authenticated by the sender. Alice and Bob should only be able to learn the number of overlaps, and the contents of the overlapping segment. This signature scheme is secure as the third party adversary can only forge a valid signature for any message with negligible probability since we use the RSA-FDH signature scheme on top of an additional step of hashing. Assuming that the hash function is a collision resistant random oracle, hashing the digest of the same hash function is collision resistant (lecture 9 slide 9). Therefore, if the RSA assumption holds and the hash function is collision resistant and irreversible, the RSA-FDH signature scheme with an additional hash step is secure. When Alice sends her hashed segments to Bob, Bob is not able to learn the contents of Alice's segments given that Bob does not have access to all the segments. In a realistic scenario, the company would have an access policy to control who has permission to which segments. Therefore, Bob will not be able to learn about the segments that do not match since the hash function is irreversible.
- Digital signature scheme using RSA-FDH
    - $(N, e, d, h_1, h_2, h_3, h_4, h_5)$ <- $Gen(1^n)$: We generate the RSA public keys and private key for the digital signature, and also the hash of the 5 segments.
    - (h, σ) <- $Sign_{sk}(h)$: The sign takes as input the digest of the segment and the secret key and signs the digest as follows:
        1. h' = H(h)
        2. $\sigma = h'^d \bmod N$
    - {1,0} <- Vrfy(h, σ) : The verification takes as input the digest and the signature and verifies the signature as follows:
        1. h' = H(h)
        2. Return 1 iff $\sigma^e == h' \bmod N$
- Third Party Adversary Goal : Forge a valid signature for the given message with non-negligible probability.