Secure Messenger Design

CY 6740 - Network Security - PS4

Team 8: San Diego (sd)

Team Members:

Ridham Bhagat (bhagat.rid@northeastern.edu)

Ritik Karayat (karayat.r@northeastern.edu)

Architecture Overview

Client-Server Model

Server:

- Clients authenticate with the server using SRP-6a for mutual authentication and key-exchange.
- After authentication, the server provides clients with necessary information (eg. public keys, IP addresses) of other clients for authenticated key establishment.

Client:

- Lightweight client that requires only a username and password.
- No storage of private keys or passwords.

Assumptions

- Since we are utilizing SRP, the (Identity, salt, verifier) of each client will be stored at the Server beforehand using an out of band secure channel.
- The hashing function used to calculate the verifier is Argon2d instead of a normal hash function like SHA256 to provide security against offline attacks. Apart from this, SHA3-512 is used for hashing everywhere else.
- Client is pre-configured with the public key of the server.
- The information provided by the server (eg. IP addresses and public keys) of other clients are trusted.
- All Symmetric Crypto operations use AES keys in GCM mode providing authenticated encryption.

Services (Security Features & Justifications)

Weak Password Protection

- SRP-6a protects against brute-force attacks against eavesdroppers since passwords are never transmitted.
- Utilizing Argon2d which is a slow hashing function and makes offline password cracking difficult for the attackers.

Denial of Service (DoS) Resistance

- Rate-limiting on 3 failed authentication attempts prevents online attacks as well.

End-Point Hiding & Identity Protection

- The identity of the clients are hidden using asymmetric encryption.
- Separate Keys for encryption and signing are used.

Perfect Forward Secrecy (PFS)

- A new key is created for each session between Client <-> Server and Peer <-> Peer to ensure PFS.

Replay / Reflection Prevention

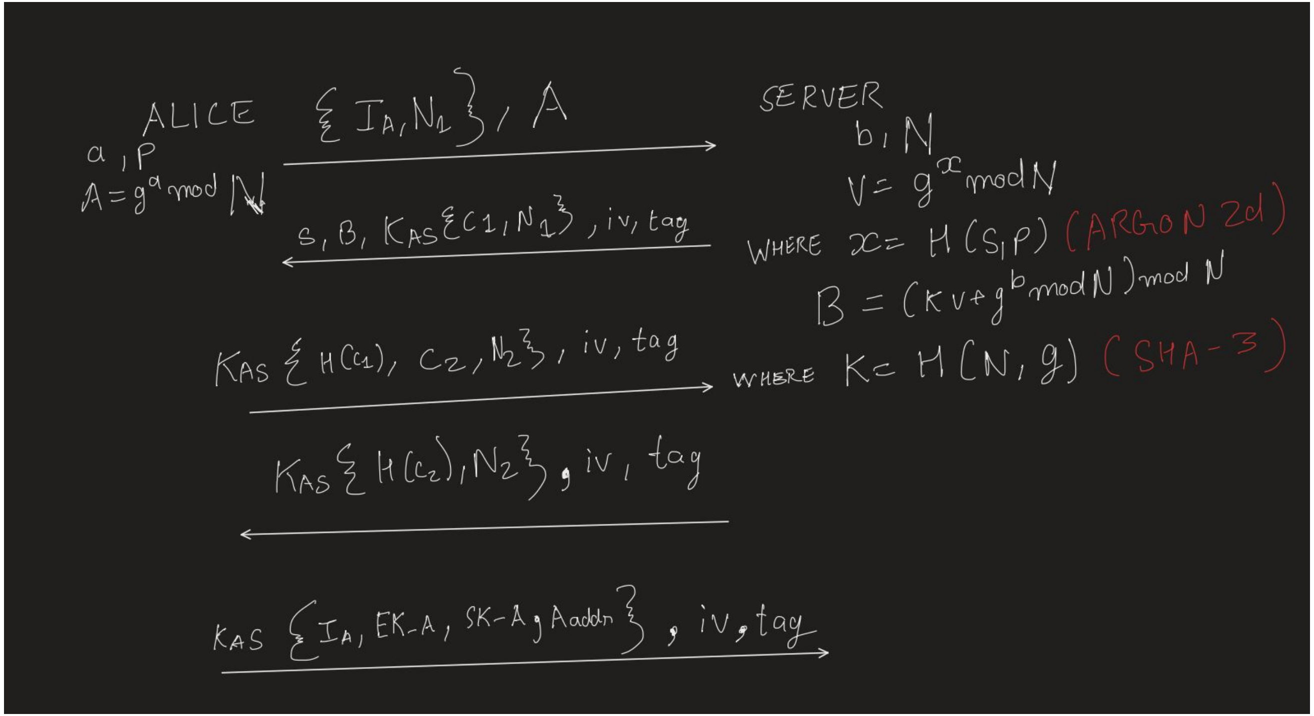
- Nonces and challenges are present to prevent replay and reflection attacks.

[Bonus] If the users do not trust the server

- We have proposed End-to-End Encryption which generates unique session keys for peer-to-peer communication not known to the server.
- Additionally we propose the implementation of TOFU (trust on first use) where users verify each others identifiers using out of band communication.

Authentication Protocol Flow (SRP - 6a)

- H() is a hash function (SHA3-512), Resistant to known plaintext length attacks
- k is a parameter derived by both sides
- s is a salt.
- I is an identifying username.
- p is the user's password.
- a & b are per session private keys.
- PKS is Public Key of the server.
- I_A is the Identity of Alice, EK_A and SK_A are public keys for encryption and verifying signatures respectively.


ALICE
a, P
$A = g^a \bmod N$
$\{I_A, N_1\}, A$
SERVER
b, N
$V = g^x \bmod N$
WHERE $x = H(S, P)$ (ARGON 2d)
$B = (KV + g^b \bmod N) \bmod N$
$s, B, K_{AS}\{C1, N_1\}, iv, tag$
$K_{AS}\{H(C_1), C_2, N_2\}, iv, tag$
WHERE $K = H(N, g)$ (SHA-3)
$K_{AS}\{H(C_2), N_2\}, iv, tag$
$K_{AS}\{I_A, EK\text{-}A, SK\text{-}A, Aaddr\}, iv, tag$

Computation of Session Key in SRP-6a

1. Alice → Server: generate random value a; send I and $A = g^a$
2. Server → Alice: generate random value b; send s and $B = kv + g^b$
3. Both: $u = H(A, B)$
4. Alice: $S_{\mathrm{Alice}} = (B - kg^x)^{(a + ux)} = (kv + g^b - kg^x)^{(a + ux)} = (kg^x - kg^x + g^b)^{(\mathrm{a + ux})} = (g^b)^{(a + ux)}$
5. Alice: $K_{\mathrm{Alice}} = H(S_{\mathrm{Alice}})$
6. Server: $S_{\mathrm{Server}} = (Av^u)^b = (g^a v^u)^b = [g^a(g^x)^u]^b = (g^{\mathrm{a + ux}})^b = (g^b)^{(\mathrm{a + ux})}$
7. Server: $K_{\mathrm{Server}} = H(S_{\mathrm{Server}}) = K_{\mathrm{Alice}}$

Communication Protocols (Post Authentication Steps)

Key Establishment Protocol:

B_{addr}= IP address of B

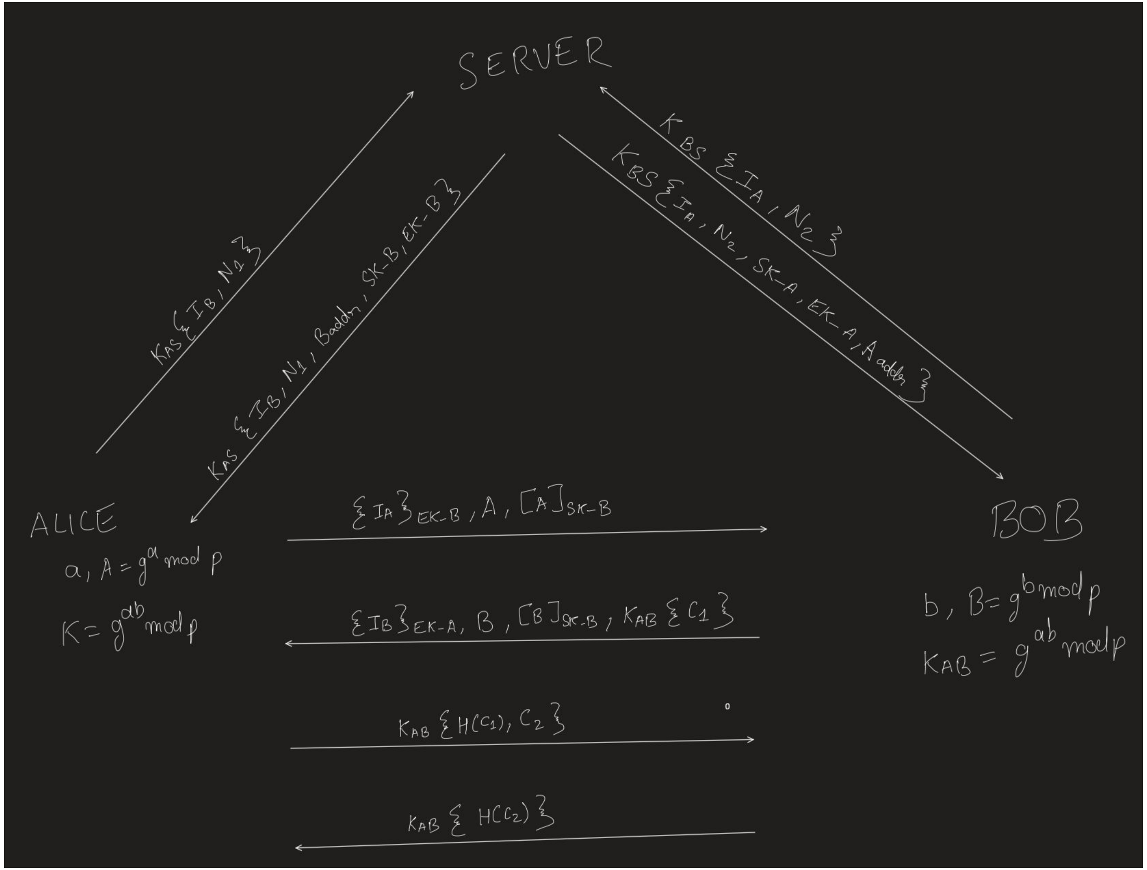
EK_*, SK_* are public keys used for encryption and verification of signatures.

Messaging Protocol:

A & B can communicate with the established key K_{AB} for the session.

Logout Protocol:

Client initiates logout procedure, it sends a logout message to the server.The server closes the TCP connection with that client and forgets the established session key. This is facilitated by TCP connection FIN along with some augmentation in the code.



SERVER
$K_{AS}\{I_B, N_1\}$
$K_{AS}\{I_B, N_1, B_{addr}, SK\_B, EK\_B\}$
$K_{BS}\{I_A, N_2\}$
$K_{BS}\{I_A, N_2, SK\_A, EK\_A, A_{addr}\}$
ALICE
$a, A = g^a \bmod p$
$K = g^{ab} \bmod p$
BOB
$b, B = g^b \bmod p$
$K_{AB} = g^{ab} \bmod p$
$\{I_A\}_{EK\text{-}B}, A, [A]_{SK\text{-}B}$
$\{I_B\}_{EK\text{-}A}, B, [B]_{SK\text{-}B}, K_{AB}\{C_1\}$
$K_{AB}\{H(C_1), C_2\}$
$K_{AB}\{H(C_2)\}$

Points of Discussion

Most of the points are discussed in the Services slide (Slide #3)

- Discuss the cases when the user trusts (vs. does not trust) the application running on his workstation. Please note that such a security guarantee might be difficult to achieve and you can focus on the first three guarantees.

  Application running on workstation compromise is an issue which can be mitigated by appropriate code signing and verifying the checksums. However, in the case if the company distributing the client is indeed malicious then the passwords of the users and chats will be compromised and the users should not use the app anymore.