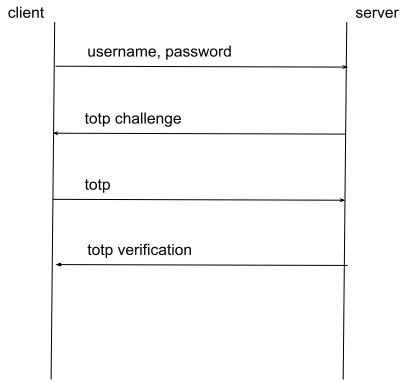
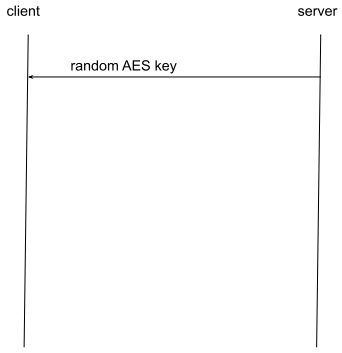
Login



For this protocol, the client sends the username and password to the server. The user has to type in those two pieces of information, and then the client will send them. This will let the server know who is joining, to be able to know which user is doing what. This includes attacks, so if a user commits one, the server would know which one did it. The server then lends the client a TOTP challenge. Then, the user types in the TOTP. This is the code that is shared only between the client and server, and the reason for using this is that attackers will not have access to the account of the user even if they know their username and password. This will reduce the risk of attackers using their account.

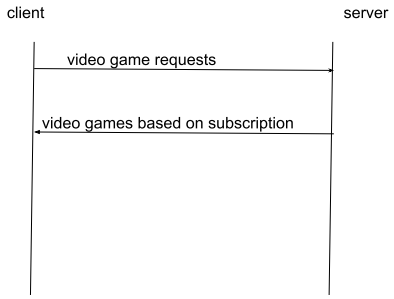
Key Exchange



Once the user is authenticated, the server generates a random AES key and sends it to the client. This key is shared only between the client and the server and will be used to encrypt all

future communication. Since AES is a symmetric encryption algorithm, it's fast and efficient, which helps keep the user experience smooth while also keeping the data safe.

Secure Session



Once the session starts, the client can send video game requests to the server. The user types in what genre of games they want, and the server responds based on that. The server also checks the user's subscription to figure out what games they have access to. Based on the subscription, it can recommend other games the user might be interested in. All of this happens securely using the AES session key.

Security Measures

Authentication: The login process (username, password, and TOTP) ensures that only the right user gets in. The password is hashed using SCrypt with a salt, which keeps it secure on the server side.

TOTP: Is the second factor, making it even harder for attackers to get in, even if they know the password.

{

"password" : "Pc9B2DvF7/uLJ/Br9xPRDy8HZnS1uDexF+WOnKTS4jA=",

"totpSecret" : "xh25qufvajoifsztneb3wrxwgaa53ffm",

"salt" : "vmiyavuYruAWPz0er4k6Hw==",

"hmacKey" : "byrQHp5lYhHOhlfwuViKJviqazfqXaL4GAl0lGRedEs=",

"username" : "alice",

"subscription status" : "subscribed"

}

Data Encryption: Everything shared between the client and server is encrypted using AES. This means no one can see sensitive data (session key, game lists, searches) unless they have the AES key available to them.

HMAC for Integrity: HMAC (Hash-based Message Authentication Code) ensures that the data hasn't been tampered with. If anything changes unexpectedly, the system can catch it because the HMAC values won't match up.

- During session, AES-GCM's built-in authentication ensures data integrity without additional HMACs.

Session Management:

Session Creation: A session is created when the user logins, binding the user to a unique session ID.

- AES session key
- Session expiry time

Logout: When the client logs out, the server deletes the AES key, ending the session. This confirms that the session can't be hijacked and that a fresh login is required next time.

Session Timeout: To prevent unauthorized access, the server automatically logs out users after a certain period of inactivity. If the client tries to interact after that time, they have to log back in.

User and Subscription Management:

User Creation: When a new user signs up, the server creates a user object containing their username, hashed password (with salt), TOTP secret, subscription status, and HMAC key. This ensures that every user is securely registered and ready for future interactions.

- Username,
- Password hash (with salt),
- TOTP secret (for MFA),
- Subscription status (e.g., "subscribed" or "not subscribed"),
- HMAC key (for verifying challenge responses).

Subscription Management: The server also handles subscriptions, so users can check their current subscription status, upgrade, or downgrade based on their preferences. This helps determine what games they have access to.

Summary:

Confidentiality (AES encryption),

Integrity (HMAC, AES-GCM),

Authentication (Password + TOTP),

Replay Attack Resistance (Nonces),

Session Security (unique keys, timeout, proper termination),

Account Security (hashed passwords, 2FA, fresh keys).