# SCRAM Authentication Protocol

Dr. Hossam Abdellatef Eng. Ahmed Ossama

**Team:**

1. Mohamed Nader 22011951
2. Abdelrahman Ihab Shafie 2206183
3. Yassin Amr Mohamed 2206190
4. Ahmed Hamed 22011938
5. Aly Waleed 22010443

1. Introduction

This project focuses on implementing the **SCRAM (Salted Challenge Response Authentication Mechanism)** protocol to ensure secure user authentication without exposing sensitive password information, even over insecure networks.

2. Objectives

 To develop a secure authentication system using the SCRAM protocol.

 To prevent common attacks such as replay attacks, password leakage, and server impersonation.

 To ensure data integrity during authentication and session establishment

3. What is SCRAM?

**SCRAM** is a password-based authentication protocol designed to address weaknesses in earlier mechanisms such as CRAM-MD5. It is defined in RFC 5802 and supports mechanisms like **SCRAM-SHA-1**, **SCRAM-SHA-256**, etc.

**Key Features:**

 Password is **never sent** over the network.

 Uses **salting** and **key derivation** (e.g., PBKDF2) for password hashing.

 Provides **mutual authentication** between client and server.

 Supports **channel binding** and optional security layers.

4. How SCRAM Works

**4.1 Pre-Authentication (Server-Side Preparation)**

1.The server stores the user’s credentials as:

* salt (random data)
* iteration count
* StoredKey = H(ClientKey)
* ServerKey = HMAC(saltedPassword, "Server Key")

**The saltedPassword is derived as:**

saltedPassword = PBKDF2(password, salt, iteration count)

**4.2 Authentication Flow**

**Step 1: Client sends first message (Client First Message)**

* Includes a random client nonce.

**Step 2: Server sends challenge (Server First Message)**

**Includes:**

* server nonce (appended to client nonce)
* salt
* iteration count

**Step 3: Client responds (Client Final Message)**

**Computes:**

* saltedPassword = PBKDF2(password, salt, iterations)
* ClientKey = HMAC(saltedPassword, "Client Key")
* StoredKey = H(ClientKey)
* AuthMessage = client-first-message-bare + "," + server-first-message + "," + client-final-message-without-proof
* ClientSignature = HMAC(StoredKey, AuthMessage)
* ClientProof = ClientKey XOR ClientSignature

**Sends:**

**c=biws,r=combinednonce,p=BASE64(ClientProof)**

**Step 4: Server verifies and responds**

* Recomputes StoredKey and verifies ClientProof.
* **If valid, responds with:**

**v=BASE64(ServerSignature)**

* Client verifies the ServerSignature for mutual authentication.

1. Security Benefits of SCRAM

* **Resistance to Replay Attacks**: Uses nonce and session-specific data.
* **Secure Password Storage**: Uses PBKDF2 with salt; resists brute-force and rainbow table attacks.
* **Mutual Authentication**: Both client and server validate each other.
* **No Plaintext Passwords**: Password is never transmitted or stored in plaintext.