Secure Software Design and Engineering
(CY-321)

Authentication Protocols

Dr. Zubair Ahmad

Authentication protocols rely on cryptographic methods to verify identity. The core principles include:

- Something You Know
- Something You Have
- Something You Are

Authentication happens between two or more parties and is the process of convincing another party that one party has indeed the identity it claims to have.

Alice and Bob want to communicate, but cant really be sure that the other is really who he/she says he/she is. So they exchange a series of messages ⇒ a protocol.


```
Alice –→ Bob : “Hi, I’m Alice.”
```

It can be extended into a mutual protocol:

```
Alice –→ Bob : “Hi, I’m Alice.”
Bob –→ Alice : “Hi, I’m Bob.”
```

The problem is of course that Eve can successfully pretend to be Alice:

```
Eve –→ Bob : “Hi, I’m Alice.
```

Usage of this Protocol

This protocol is actually in widespread use:

Telephone calls are usually not (properly) authenticated; otherwise

May be one-sided: Alice may be a computer and Bob may be a user. Bob logs in to Alice; Alice then knows its Bob, but Bob doesnt (in general) know its Alice.

May be mutual: Bob logs in to Alice so that both of them are convinced of the other' s identity afterwards.

May use trusted third parties, online (Bob asks the trusted party—Trent—to establish a conversation with Alice) or offline (Alice could present a certificate signed by Trent).

There might be an eavesdropper—Eve—that can listen to and/or modify messages as they are exchanged between Alice and Bob.

There might be an **intruder**—Trudy—that can listen to and inject messages.

Threats Against Authentication Protocols

The basic threat is always that it is possible for Trudy or Eve eventually to impersonate Alice or Bob. They can accomplish this for example by

- Replaying all or part of a previously recorded conversation;
- Eavesdropping on a conversation and learning secrets;
- Modifying messages en route to their destination;
- Modifying the message flow by inserting or deleting messages in the network.
- Assuming another identity (e.g., using the othe network address).
- Stealing another databases, to steal keys.

How can this protocol be improved?

Alice and Bob could share a secret. Alice could present that secret to show that she really is Alice. (Who you are is what you know.)

Variation: Alice claims that she knows a secret that is unique to her. Instead of presenting the secret, Alice could prove that she knows the secret without divulging it (**zero-knowledge-proof**).

How can this protocol be improved?

Alice could be in the possession of a unique token that she presents to Bob. (Who you are is what you have.)

Alice could agree on submitting to a biometric scan, e.g., a fingerprint scan or face scan. (Who you are is what you are.)


```
Alice –→ Bob : N, {M, N}K
```

This notation means that the principal Alice transmits to the principal Bob a message containing a nonce N, and the plaintext M concatenated with N, encrypted under the key K.

A nonce is anything that guarantees the freshness of a message, such as a random number, a serial number, or a challenge received from a third party.

Nonces

As we have said, a nonce is a number used only once. It is possible to introduce weaknesses into protocols if the nonces have the wrong properties.

Nonce types are:

- a timestamp;
- a sequence number; and
- a large random number

Large Random Numbers as Nonces

Why can we use a random number as a nonce when there is a chance that it would be reused?

Large Random Numbers as Nonces

We want to find out **how likely it is to generate duplicate random numbers** when picking from a huge pool of possible values.

We are **generating random numbers** (nonces).

Each number is **128 bits long**.

There are **2^128** possible unique values.

We generate **one nonce every millisecond** for **1000 years**

.We need to check: **Will we get a duplicate nonce?**

Step 1: Understanding the Problem

We want to calculate the probability that **k random values** selected from a set of **N unique values** are **all different** (i.e., no duplicates).
Each value is chosen **randomly and independently**.

Step 2: Defining Notation

N = total number of possible values (e.g., for a 128-bit random number, N=2^128

k = number of values we randomly generate (e.g., the number of nonces we generate).

•The probability that all k values are **unique** is denoted as P.

Step 3: Writing the Exact Probability Formula

Lets Do it

Step 4: Approximation Using δ

Computing this exact formula for large N is difficult. So, we approximate it using a small relative change **δ**.

Defining δ

The difference between the total values N and the last available value N−k+1 is:

$$\delta = k - 1 \backslash N$$

which represents the **fractional reduction** in available choices.

Using this, we approximate:

$$N - k + 1 \approx (1 - \delta)N$$

Step 5: Applying the Approximation to the Probability Formula

We substitute $N-k+1 \approx (1-\delta)N$ into the numerator:

$$N(N-1)\cdots(N-k+1) \approx N^k(1-\delta)^k$$

$$P = N^k(1-\delta)^k / N^k$$

$$P \approx (1-\delta)^k$$

Password-Based Authentication Protocols

These protocols rely on a shared secret (e.g., passwords) between a user and the system.

1. One of the simplest authentication mechanisms.
2. The client sends a username and password in plaintext.
3. **Security Issue:** Susceptible to eavesdropping and man-in-the-middle attacks.

Challenge-Handshake Authentication Protocol (CHAP)

More secure than PAP.
Uses a challenge-response mechanism:

- Server sends a random challenge.
- Client hashes the password with the challenge and returns the result.
- Server verifies the hash.

Advantage: Prevents replay attacks since challenges are different each time.

Token-Based Authentication Protocols

Kerberos

Uses a **ticket-based authentication system**.
Works in a trusted **Key Distribution Center (KDC)** that issues tickets.
Uses symmetric encryption (e.g., AES) for secure communication.

Process:

- User authenticates to the KDC.
- KDC issues a Ticket Granting Ticket (TGT).
- User requests access to a service, presenting the TGT.
- KDC provides a session key for secure communication with the service.

Token-Based Authentication Protocols

OAuth 2.0

- Designed for web and mobile applications.
- Allows users to grant third-party applications limited access to their resources without sharing passwords.
- Uses access tokens and refresh tokens.
- **Common in:** Google, Facebook, and API-based authentication

Token-Based Authentication Protocols

OpenID Connect (OIDC)

- Built on top of OAuth 2.0.
- Adds identity verification features.
- Provides user authentication using ID tokens.
- **Used in:** Single Sign-On (SSO) systems.

Public Key-Based Authentication Protocols

Secure Shell (SSH) Authentication

- Uses **public-key cryptography** instead of passwords.
- The user generates a public-private key pair.
- The public key is stored on the server, and the private key remains with the user.
- **Advantage:** No password transmission, making it resistant to brute force and phishing attacks.

Public Key-Based Authentication Protocols

Transport Layer Security (TLS)

- Used in HTTPS for secure web communications.
- Uses a **handshake mechanism** to establish an encrypted session.
- Supports mutual authentication using **client and server certificates**.
- **Used in:** Web browsers, email security, VPNs.

Public Key-Based Authentication Protocols

RADIUS (Remote Authentication Dial-In User Service)

- Centralized authentication for remote access.
- Supports **username/password, token-based, and certificate-based authentication**.
- Often used in **Wi-Fi security and VPNs**.

Biometric Authentication Protocols

FIDO2 (Fast Identity Online)

1. Uses public key cryptography and biometric authentication.
2. Eliminates passwords using **WebAuthn and CTAP (Client-to-Authenticator Protocol)**.
3. **Used in:** Passwordless authentication (Windows Hello, Apple Face ID)

Questions??

zubair.ahmad@giki.edu.pk

Office: G14 FCSE lobby