# 6.1 Question 1

### 6.1.A List the three main forms of authentication. Which one can be the most secure? Why?

- Password-Based Authentication
- Address-Based Authentication
- Cryptographic Authentication

The most secure form would be the Cryptographic Approach using Two-Factor Authentication, as an attacker first needs to find the correct password (at least 8 characters with all 26 letters and special symbols possible) and then get a hand onto the second device, like a smartphone, in order to successfully pass the Two-Factor Authentication Protocol. This certificate can also be secured with another password or even biometric information like a fingerprint for example, which can get really hard for an attacker to pass.

### 6.1.B An old website is storing its user's passwords inplain text. You are hired to enhance its security. How would you store and use the passwords (and why)? What type of password attacks are there? What would your counter-measures for them be?

I would implement the use of an Authentication Storage Node, which adds a new, additional layer of security. The server itself needs to authenticate itself when accessing the ASN in order to get access to the stored passwords. To further add security the plaintext passwords can be stored in their encrypted form.
The following attacks would be possible:

- **Offline Attack**
  Attacker learns the encrypted password and can find the corresponding password without needing to send requests to the website. In order to counter-measure against these attacks the password space must be as large as possible in order to make the search for an encrypted password infeasible. Also, the user can be asked to change the password frequently.
- **Online Attack**
  Attacker tries a random password until it is accepted. In order to resist to such attacks the amount of password tries can be limited or add a time-limiter between tries. Also the use of longer passwords can ensure a higher level of security against such attacks.

### 6.1.C A website is using one common salt for all user's passwords, along with server-side hashing. Would using a different salt per user improve the security (and why)? Does a longer salt provide better security (and why)? Would you recommend (or not) to store the salts in an encrypted form (and why)? Assuming no salt was used, would client-side hashing (instead of server-side hashing) decrease the security (and why)?

When using salt the need of any rainbow-table is not required. The major disadvantage of using a common salt is that if an attacker learns it, he can easily determine the password of other users, hence to add security a different salt for each user should be used.
Because a longer salt leads to a much greater searchable space in order to successfully bruteforce attack, hence making such attacks much more infeasible, leading to a higher security.
As salts are already random values used for encrypting data there is really no point to store these salts in the encrypted form.
Yes, without the use of salt the security of the transfer of data would be decreased as an attacker "only" needs to eavesdrop the sent message and determine the encrypted password. Then the attacker can

perform a bruteforce attack very easily in order to determine the solution using the password and the corresponding hash.

### 6.1.D A website is sending "plaintext" passwords over HTTPS. Would the website be safer if these passwords were encrypted (and why)?

Encrypting the "plaintext" password before it is sent over HTTPS wouldn't make the website safer. This is because the encrypted password could be simply used as it would be the actual password, as the server wouldn't know the difference.