IDATT2503 - Cryptography Assigment 6

Edvard Berdal Eek

November 2025

Task 1

1. **What are the two main types of password attacks?**

   It was not very clear what answer this question was looking for, as the lecture specifies two types of brute force password attacks, online and offline, not general password attacks.

   Online attacks is where attackers use the same authentication interface as the user. Here they are limited by rate limiting, lockouts etc.

   Offline attacks is where the attacker has obtained the hashed passwords from for example a leak. Here the attacker can use unlimited attempts to try and reveal the plain text password.

   If the question did not look for these two types of attacks, it could for example be brute force and dictionary attacks or more general ones like automated (brute force, dictionary) and human-focused attacks (phising).

2. **Why should passwords not be stored encrypted?**

   Encrypted passwords are more vulnerable when compromised than hashed and salted passwords. A leaked encryption key would reveal the plain text passwords if the encrypted passwords were compromised. In addition, the encrypted passwords would reveal the plain text length if compromised.

3. **What is the advantage to use a specialized password algorithms over general cryptographic hash functions using a salt?**

   The lecture lists three reasons why traditional hash functions are more vulnerable than specialized password algorithms. Hash functions like SHA-256 are very fast to compute, require minimal memory and is therefore vulnerable to brute force attacks on specialized hardware.

   Memory-intensive hash functions like **scrypt** and the recommended **Argon2** force attackers to allocate large amounts of memory per attempt, making attacks expensive by requiring more resources.

Task 2

Let $p = 97$ and let $g = 5$ be a generator modulo 97.

1. **Choose the private key $x = 13 (= (1101)_2)$ and compute the public key $k = g^x \bmod p$.**

```
1. Compute the public key k = g^x mod p
k = 5^13 mod 97 = 29
Public key: (p = 97, g = 5, k = 29)
```

Figure 1: Public key

2. **Encrypt the message $m = 42$ using the (random) ephemeral key $y = 19 (= (10001)_2)$ to obtain the ciphertext (c1, c2).**

```
2. Encrypt the message m = 42 using the ephemeral key y = 19 to obtain the ciphertext (c1, c2)
c1 = g^y mod p: 5^19 mod 97 = 38
c2 = m * k^y mod p: 42 * 29^19 mod 97 = 66
Ciphertext: (c1 = 38, c2 = 66)
```

Figure 2: Ciphertext

3. **Decrypt the ciphertext using the private key $x = 13$.**

```
3. Decrypt the ciphertext using the private key x = 13.
s = c1^x mod p: 38^13 mod 97 = 57
s^-1 = 80 der s * s^-1 mod p = 1: 57 * 80 mod 97 = 1
m_d = c2 * s^-1 mod p: 66 * 80 mod 97 = 42
```

Figure 3: Decrypted ciphertext

4. **When recovering m, compute the modular inverse using the Extended Euclidean Algorithm and show all steps.**

   I computed the modular inverse m^{-1} using the Extended Euclidean Algorithm python implementation provided in lecture 4, only tweaked to output all the steps.

```
4. When recovering m, compute the modular inverse using the Extended Euclidean Algorithm and show all steps.
Compute: a=42, b=97, q=0, r=42
 Compute: a=97, b=42, q=2, r=13
  Compute: a=42, b=13, q=3, r=3
   Compute: a=13, b=3, q=4, r=1
    Compute: a=3, b=1, q=3, r=0
     Base case: gcd=1, coefficients (1,0)
    Backtrack: gcd=1, w=0, z - q * w=1  (check: 0*3 + 1*1 = 1)
   Backtrack: gcd=1, w=1, z - q * w=-4  (check: 1*13 + -4*3 = 1)
  Backtrack: gcd=1, w=-4, z - q * w=13  (check: -4*42 + 13*13 = 1)
 Backtrack: gcd=1, w=13, z - q * w=-30  (check: 13*97 + -30*42 = 1)
Backtrack: gcd=1, w=-30, z - q * w=13  (check: -30*42 + 13*97 = 1)

m^-1 = -30 where m * m^-1 mod p = 1: 42 * -30 mod 97 = 1
c2_tampered = c2 * n * m_inverse: 66 * 2 * -30 = -3960
m_d = c2_tampered * s^-1 mod p: -3960 * 80 mod 97 = 2
```

Figure 4: Euclidean Extended to find m^{-1}, and altered m_d

With the modular inverse of m, m^{-1}, an attacker can alter the ciphertext c_2 to make the recipients decrypted message be any given message n.

Task 3

1. **What does it mean that ElGamal is "malleable"?**

ElGamal is called malleable because it is multiplicatively homomorphic, allowing attackers to tamper with the ciphertext in a controlled way, such that the plaintext produced after decryption changes in a predictable way.

2. **Describe how Mallory could tamper with the encrypted message so that the decrypted result becomes double the original amount**

   Because ElGamal is multiplicatively homomorphic if c_2 is multiplied with a value n the resulting m will be a factor of n.

   $$\textbf{Cipher text: } (c_1, n \cdot c_2) \rightarrow \textbf{Plain text: } n \cdot m$$

   So if Mallory just multiplies c_2 with 2 the decrypted result will be doubled.

3. **Why is this dangerous?**

   This is dangerous as it would produce seemingly authentic messages, despite being tampered with. If this was applied to for instance banking transactions, an attacker could multiply the encrypted transaction by 10 would make the decrypted transaction 10 times the original transaction.

   If the attacker also knew the plain text message m, they can find it's multiplicative inverse m^{-1} and use it to make the result be any message n they want. Altering the cipher text to $(c_1, n \cdot m^{-1} \cdot c_2)$ would make the decrypted plain text message be n.