Aufgabe 1)

b)
Assmannn:
Login: 5518
Entry: IT Security; is exciting

Koohiana:
Login: 5441
Entry: IT Security; is exciting

Aufgabe 2)

a) The transformation rounds are a convinient protection against brute force attacks. Since the effectiveness
of brute force attacks depend on the time consumed per try, the "defender" can easily raise the transformation
rounds in order to increase the time.

b) Yes, the ECB Mode is weakening the overall transformation. ECB Mode encrypts similar parts of the
plaintext equaly such that the ciphertext is going to be similar on the respective parts as well. That means the attacker can retrieve valuable information about the key used. In the KeePass scenario
an attacker could use this weakness by using the XML Tags as similar palintext parts, the XML tags gonna
be similar in every user database. An attacker could test encrypt many databases in the same way and find
out, which parts stay similar after the EBC Mode encryption.

c) Average time for 10000 digits = 3800 ms -> 2632 passwords per second.

For 10000 repetition rounds:
1. Extending key possiblities including 26 minor characters of the standard alphabet -> (26 + 10)^4 = 1679616
Time: 1679616/2632 = 638seconds

2. Extending key possiblities including 26 minor and 26 captial characters of the standard alphabet -> (26 + 26 + 10)^4 = 14776336
Time: 14776336/2632 = 5614seconds

For 1000000 repetition rounds:
Average time for 10000 four digit variations: 253968ms ~ 39 passwords per second

1. Extending key possiblities including 26 minor characters of the standard alphabet -> (26 + 10)^4 = 1679616
Time: 1679616/39 = 43067 seconds = 718 minutes ~ 12 hours

2. Extending key possiblities including 26 minor and 26 captial characters of the standard alphabet -> (26 + 26 + 10)^4 = 14776336
Time: 14776336/39 = 378880 seconds ~ 6315 minutes ~ 105 hours ~ 4,5 days

d)
A KDBX file consists of multiple parts, starting with the Header. The Header contains the stream start bytes, located in the first 32 unencrypted bytes of the file.

If the integrity of the stream start bytes is compromised, the Keepass software will not recognize the file as secure and cannot open it. The stream start bytes serve to validate the file's integrity and the correctness of the key.

The 32-byte stream start bytes should have been randomly generated during file saving.

The database section of the file immediately follows the stream start bytes. To validate the generated final master key, compare the initial 32 bytes with the stored "Stream Start Bytes" field.

The data in the file consists of multiple blocks, with each block structured as follows:

Block ID: An integer representing the block's ID. The first block has an ID of 0.
Block Hash: A 32-byte hash value of the Block Data.
Data Size: An integer indicating the size of the Block Data.
Block Data: The actual data contained within the block.
By analyzing and comparing these block structures, you can effectively process the KDBX file.

Utilizing cryptographic hash functions provides a reliable method to ensure file integrity in .kdbx files without explicitly storing the first 32 bytes in plaintext. By computing and comparing hash values, the integrity of the file can be verified, offering robust protection against tampering. Implementing industry best practices and using secure hash functions are essential for effective file integrity checks.