Peer-to-peer networks

Exercise 2.2: Hash-values:

1) Why are hash-values used in content distribution networks?

Solution: They are used as identifiers for files. The hash should be unique for the content of the file, the filename is not.

2) You are offered a requested file from an anonymous participant in a content distribution network. The name of the offered file matches your request but the MD5 hash differs. Would you accept? Or would you accept a matching MD5 hash but a different filename? Explain why. (The hash can be verified by you after download and is thus considered to be credible.)

Solution: Name matches, MD5 sum dissimilar: The name can be faked easily. It would be trivial for an attacker to rename a Trojan horse to the name of the requested filename. In fact, this happens if an arbitrary filename is searched in Gnutella (due to malicious servers and not due to the protocol, of course).

Name mismatches, MD5 similar: If you trust the MD5 sum, you can safely download the file, calculate the MD5 sum yourself and compare. But note: An attacker can offer any file with the correct checksum and you can't be sure about the file's content unless you have a trusted directory or a large number of participant offer the same file.

Peer-to-peer networks

Exercise 2.2: Hash-values:

3) Excursus: One ability for hashes is to hide clear texts. Many (operating) systems store passwords as hash values rather than as clear-text. A password entered for login is hashed and compared to previously hashed passwords in a password file. If the hashes are equal, access is granted.

Advantage: Even if an attacker can get hold of the password file she can not access a system from outside without knowing which password results in a certain hash. Entering the hash-value will not be successful, of course. But recently, trivial method have become well-known, how the clear-text behind in particular simple password can be found.
Find a way to get the original password which results in the following MD5 hash:

380e537acdaedd487ca1adb49d020f7e

Solution: Enter the hash in google and get the password “**overlay**“ as clear text.

Peer-to-peer networks

Exercise 2.3:

A sender wants to transmit the following 32 bits

```
10110100 01011011 01010101 10110110
```

in four chunk-packets, each of which contains 8 bits. Both, sender and receiver use the same random number generation which produces the following bit-stream:

```
1110 0101 1001 0110
```

For data transmission, the **Random Linear Fountain Code** from the lecture is used.

Proceeding:	- Divide the message into chunks.
Sender Side	Combine the chunks bit-wise according to the bit-merging vector which is taken from the output of the random number generator.
	- Stop sending further chunks as soon as a sufficient number of XORed chunks with linear independent merging vectors have been sent.
Receiver Side	- Collect the incoming chunks until a sufficient number is received. Sufficient means, that their merging vectors are linearly independent. The merging vectors are taken from the output of the random number generator as was done on the sender side.
	- After having gathered enough data, calculate the modulo 2 inverse matrix of the matrix formed by the merging vectors.
	- XOR the received chunks according to the inverse matrix.

Peer-to-peer networks

Exercise 2.3:

A sender wants to transmit the following 32 bits: **10110100 01011011 01010101 10110110**

```
1110 10110100      10111010 (transmitted)
0101 01011011      11101101 (transmitted)
1001 01010101   =  00000010 (transmitted)
0110 10110110      00001110 (transmitted)
```
  ˆ- Taken from the random source

Calculate inverse matrix on receiver side

```
1110 1000 <-+     --+          ...(continued)
0101 0100   | --+   |
1001 0010   |   | <-+          1000 1001
0110 0001 --+ <-+              0010 1110 --+
---------                      0100 1111   |
1000 1001                      0011 0101 <-+
0101 0100     <-+              ---------
0111 1010 <-+ --+              1000 1001
0011 0101 --+                  0100 1111
---------                      0010 1110
...                            0001 1011
```

Peer-to-peer networks

Exercise 2.3:

On the receiver side, the inversion worked and the received packets can be transformed into the original clear text:

```
1001 10111010   10110100
1111 11101101 = 01011011
1110 00000010   01010101
1011 00001110   10110110
```