Secure and Dependable Systems Module: CO-566
Constructor University Date: 2023-04-14
Dr. Jürgen Schönwälder **Due: 2023-04-21**

Problem Sheet #9

Problem 9.1: *eavesdropping on rsa* (4+1 = 5 points)

Alice is sending Bob a secret RSA-encrypted message. Bob has published his public RSA key $k = (e, n) = (1739959, 8305897)$. Eve managed to obtain a copy of the secret message. Eve recorded the following sequence of decimal numbers:

```
2960611,5203400,1366829,5919701,567261,5812140,7301975,5144352,
3467384,7301975,6157330,5203400,1366829,5919701,567261,5812140,
84215,7301975,1561607,1366829,2921766,1366829,5203400,4166410,
7301975,7797451,5144352,2921766,5919701,3467384,3837045
```

a) Help Eve to decrypt the numbers. Explain the steps you are doing.

b) Assuming the decrypted numbers are character code points, what was Alice's message to Bob?

Problem 9.2: *diffie hellman key exchange* (1+2 = 3 points)

Alice and Bob agree on using the prime number $p = 191$ and the primitive root $g = 42$. Alice randomly chooses the value $a = 27$.

a) Which value does Alice send to Bob?

b) After the key exchange, Alice has the key $k = 178$. Which value did Bob choose and which value did Bob send to Alice?

Problem 9.3: *proof of work* (1+1 = 2 points)

Cryptographic hash functions can be used for a proof of work, also known as a cryptographic puzzle. The challenge is to find a random value that appended to a given message causes the the hash value to have a certain format, e.g., N leading bits of 0.

a) Find a random sequence of 64 hexadecimal digits (different from the one on this sheet) such that the SHA-256 checksum begins with 12 bits (three digits in hexadecimal notation) of 0s. (Since your result is a random solution, we expect it to be different from the results produced by other students.)

   We will test your solution using `openssl sha256`. More precisely, we will use:

   ```
   m=e9d90603ede2b22e8714dfa340a2911079431c91ab4d55a412a64a6ba4593bc2
   /bin/echo -n $m | openssl sha256 -r
   ```

b) Provide a script (python, shell, haskell, ...) that searches for a solution of the puzzle. Make sure your script can be run by us and that it is understandable.