Exercise Sheet Cryptanalysis

Solve the following exercises during class.
Background: You have an HTML document with the following content at hand:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Signed contract</title>
   </head>
   <body>
      I, Alice, transfer an amount of 100 Euros to Bob.
   </body>
</html>
```

This document has been digitally signed by Alice.

LB-CA 00. (<u>not</u> to be submitted)

a) Compute the signed SHA-256 hash of this file by using your program from LB-S 01. Subsequently, check the integrity of the file by verifying the signature.

b) Modify the content of the file such that Alice transfers an amount of *200 Euros* to Bob. Now, check the integrity of the file again. Is the signature still valid?

LB-CA 01. (<u>not</u> to be submitted)

(*Code::Blocks* template *libsodium project*)

In the original document, Alice transfers an amount of 100 Euros to Bob. A "fake" document is to be created in which Alice transfers an amount of 200 Euros to Bob. However, this fake document is supposed to have the exact same signature as the original document (*second-preimage attack*). By making changes to the markup which are small, but invisible for a viewer, try to achieve an identical hash value and thus an identical signature as the original document. To do so, write a suitable program for modifying the markup and computing the hash values. Finally, the program is supposed to output a variant of the document with an identical hash.
Important: SHA-256 has 2^{256} possible, distinct output values. Therefore, on average, 2^{255} attempts are necessary in order to produce an identical hash with a second-preimage attack. Thus, use an extremely simplified (and practically insecure) output variant which only uses the first four hexadecimal digits of the hash value. This reduces the average number of required attempts.

LB-CA 02. (not to be submitted)

(*Code::Blocks* template *libsodium project*)

Now, try to create two documents – one "original document" (100 Euros Alice to Bob) and a "fake document" (200 Euros Alice to Bob), where both documents have the same hash value (*collision attack*). Proceed analogously to exercise LB-CA 01. and create the collision through changes to the markup of **both documents** which are small, but invisible for a viewer. First, reflect on what the difference to the attack above is and how it affects the average number of attempts. Again, write a suitable program for modifying the markup of **both documents** and for computing the hash values. Finally, the program is supposed to output one variant for each of the two documents with an identical hash. When checking for collisions, make sure that **all** variants of the generated documents are considered.

Hint: Instead of comparing the hash of every original document variant to that of every fake document variant – which has quadratic complexity in the number of variants –, use sets (`std::set` or `std::multiset`) or maps (`std::map` or `std::multimap`) to collect the data and to reduce the complexity required for comparisons and checks.