Aufgabenblatt Kryptoanalyse

Lösen Sie die nachfolgenden Aufgaben in der Lehrveranstaltung.
Ausgangssituation: Sie haben ein HTML-Dokument mit folgendem Inhalt vorliegen:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Signierter Vertrag</title>
   </head>
   <body>
      Ich, Alice, überweise Bob den Betrag von 100 Euro.
   </body>
</html>
```

Dieses Dokument wurde von Alice digital signiert.

LB-KA 00. (nicht abzugeben)

a) Berechnen Sie den signierten SHA-256-Hash dieser Datei, indem Sie Ihr Programm aus LB-S 01. verwenden. Prüfen Sie anschließend die Integrität der Datei, indem Sie die Signatur verifizieren.

b) Verändern Sie den Inhalt der Datei dahingehend, dass Alice den Betrag von *200 Euro* an Bob überweist. Prüfen Sie nun erneut die Integrität der Datei. Ist die Signatur noch gültig?

LB-KA 01. (nicht abzugeben)

(*Code::Blocks*-Template *libsodium project*)

Im Originaldokument überweist Alice an Bob den Betrag von 100 Euro. Es soll ein „gefälschtes“ Dokument erzeugt werden, in welchem Alice den Betrag von 200 Euro an Bob überweist. Dieses gefälschte Dokument soll jedoch die exakt gleiche Signatur wie das Originaldokument aufweisen (Zweites-Urbild-Angriff, engl. *second-preimage attack*). Versuchen Sie durch kleine, aber für den Betrachter nicht sichtbare Änderungen im Markup einen identischen Hashwert und somit eine identische Signatur wie das Originaldokument zu erreichen. Schreiben Sie dafür ein geeignetes Programm zum Modifizieren des Markups und zum Berechnen des Hash-Wertes. Das Programm soll abschließend eine Variante des Dokuments mit identischem Hash ausgeben.
Wichtig: SHA-256 hat als Ausgabe 2^{256} mögliche verschiedene Werte, wodurch im Mittel 2^{255} Versuche notwendig sind, um einen identischen Hash bei einem Zweites-Urbild-Angriff zu erzeugen. Verwenden Sie für die Aufgabe daher eine stark vereinfachte (und in der Praxis unsichere) Variante, bei der nur die ersten vier Hexadezimalziffern des Hash-Wertes verwendet werden. Dies reduziert die im Mittel notwendigen Versuche.

LB-KA 02. (nicht abzugeben)

(*Code::Blocks*-Template *libsodium project*)

Versuchen Sie nun zwei Dokumente zu erzeugen, ein „Originaldokument“ (Alice an Bob 100 Euro) und ein „gefälschtes Dokument“ (Alice an Bob 200 Euro), die beide den gleichen Hashwert haben (Kollisionsangriff, engl. *collision attack*). Gehen Sie dabei analog zur Aufgabe LB-KA 01. vor und erzeugen Sie durch kleine, aber für den Betrachter nicht sichtbare Änderungen im Markup **beider Dokumente** die Kollision. Überlegen Sie zunächst, worin der Unterschied zum obigen Angriff liegt und wie sich dieser auf die Anzahl der im Mittel notwendigen Versuche auswirkt. Schreiben Sie dafür wiederum ein geeignetes Programm zum Modifizieren des Markups **beider Dokumente** und zum Berechnen der Hash-Werte. Das Programm soll abschließend je eine Variante der beiden Dokumente mit identischem Hash ausgeben. Stellen Sie bei der Überprüfung auf Kollisionen sicher, dass **alle** Varianten der generierten Dokumente miteinbezogen werden.
Hinweis: Anstatt den Hash jeder Originaldokumentvariante mit dem jeder gefälschten Dokumentvariante zu vergleichen – was eine quadratische Komplexität in der Anzahl der Varianten hat –, verwenden Sie Mengen (`std::set` oder `std::multiset`) oder Maps (`std::map` oder `std::multimap`), um die Daten zu sammeln und die für Vergleiche und Überprüfungen notwendige Komplexität zu verringern.