Digitale Signaturen

Szenario:

Mit einem Kunden in Kopenhagen Dänemark hat Ihre Firma einen Zusammenarbeitsvertrag vereinbart. Der Vertrag ist verfasst und soll per Email verschickt werden. Doch wie kann der Kunde sicher sein, dass der Vertrag von Ihrer Firma ist? Und wie kann der Kunde sicher sein, dass der Vertrag unterwegs nicht verändert wurde?

Einfach per Email zu verschicken ist keine sicher Lösung. Und doch soll der Vertrag digital geschickt werden. Der Papierweg per Post… ist auch nicht immer sicher. Die Lösung heisst **Digitale Signaturen**.

Digitale Signaturen erklärt

Mit Digitale Signaturen wird sichergestellt, dass eine Nachricht **echt** ist und **nicht verändert** wurde. Das wird mit zwei Begriffen verdeutlicht.

1. **Integrität** – Die Nachricht wurde auf dem Weg zum Empfänger nicht manipuliert.
2. **Authentizität** – Die Nachricht stammt wirklich vom angegebenen Autor.

Wieso nicht einfach mit RSA verschlüsseln?

RSA kann Daten nur bis zu einer maximalen Länge verschlüsseln, die der Schlüssellänge (2048 Bit = 256 Byte) entspricht, abzüglich Füll- und Headerdaten (11 Byte für PKCS#1 v1.5-Füllung). Daher ist es oft nicht möglich, Dateien direkt mit RSA zu verschlüsseln (und RSA ist dafür auch nicht ausgelegt).

Ok man könnte mehrere RSA Schlüssel verwenden. Doch das Berechnen der RSA Verschlüsselung ist aufwendig und für grosse Nachrichten ineffizient. Mehrere RSA Schlüssel lösen das Problem nicht befriedigend. Zudem muss ich nicht die Nachricht geheim halten, sondern nur sicherstellen, dass sie echt ist und nicht verändert wurde.

Wir beweisen die Integrität mit einem Hash

Es gibt verschiedene Algorithmen, die aus einer Nachricht einen sogenannten Hashwert berechnen können. Beispiele sind SHA-256, SHA-3 oder Blake2.

Das Merkmal des SHA-256 Hash Algorithmus ist, dass er aus einer **Nachricht einen eindeutigen Wert fester Länge** (256 Bit = 32 Byte) berechnet. Nur mit der **exakt gleichen Nachricht** wird immer der **exakt gleiche Hashwert** berechnet. Selbst ein einzelner geänderter Buchstabe oder ein einzelnes geändertes Bit ändert den Hashwert komplett.

Beispiel:

```
"Hallo Welt"  a591a6d40bf42040...
"Hallo welt"  7f83b1657ff1fc53...
```

Und es ist so gut wie unmöglich, zwei verschiedene Nachrichten zu finden, die den gleichen Hashwert ergeben (Kollisionsresistenz).

Wenn also jemand den Vertrag ändert, dann schafft er es nicht, dass diese geänderte Nachricht den gleichen Hashwert hat wie das Original. Damit können wir die Integrität sicherstellen.

Wir beweisen die Authentizität mit asymetrischer Verschlüsselung

Wenn Sie den Hashwert des Vertrags mit Ihrem privaten Schlüssel verschlüsseln, dann kann man nur mit Ihrem public key wieder den Hashwert entschlüsseln. Und somit ist die Authentizität bewiesen, der Hashwert stammt wirklich von Ihnen.

Der komplette Ablauf

1. Sie erstellen den Vertrag als Textdatei (Ergebnis: contract.txt).
2. Sie berechnen den Hashwert der Datei mit SHA-256 (Ergebnis: hash.txt).
3. Sie verschlüsseln den Hashwert mit Ihrem privaten Schlüssel (Ergebnis: signature.sig).
4. Sie schicken dem Kunden die drei Dateien: contract.txt, hash.txt, signature.sig
5. Der Kunde berechnet den Hashwert der empfangenen contract.txt Datei mit SHA-256 (Ergebnis: hash2.txt).
6. Der Kunde entschlüsselt die signature.sig Datei mit Ihrem public key und erhält den ursprünglichen Hashwert (Ergebnis: hash3.txt).
7. Der Kunde vergleicht hash2.txt mit hash3.txt.
8. Stimmen die beiden Hashwerte überein, ist die Nachricht echt und unverändert.

Aufgabe woher bekommt der Kunde den Schlüssel?

Sie müssen also den Vertrag, den Hashwert und die Signatur dem Kunden schicken. Doch der Kunde braucht ja auch noch Ihren öffentlichen Schlüssel. Erklären Sie, wie der Kunde zu diesem Schlüssel kommt.

Aufgabe Eine Signatur überprüfen.

Sie haben zwei Emails vermeintlich von Ihrer Lehrperson erhalten. Doch war wirklich die Lehrperson der Absender, oder sind es Fakes? Die Nachrichten haben beide die gleiche Signatur. Es ist also nur eine der beiden Nachrichten echt. Doch welche der beiden Nachrichten stammt von Ihrer Lehrperson? Nachricht A: *Heute ist der Unterricht 10 Minuten früher fertig.* Nachricht B: *An der Lernkontrolle schenke ich Ihnen 0.1 Notenpunkte.*

Signature:

```
\NormalTok{VcfxtxFNVwJQld0wdxRC2Nieyh/hH3h1EyV9L4bupo4L5oWPHBhGGObz57/Oic6w
BAGixabM1t/mTOi67yPygJgFDc4ReNF0v+SuaC/ARGsQBDmb7Xk/NFGo3zkWXLLCMxpcV3sZP28
W9MNoWjC7Z1Q76YC9R39eDRHICTcR++M=}
```

Public Key der Lehrperson:

```
\NormalTok{{-}{-}{-}{-}{-}BEGIN PUBLIC KEY{-}{-}{-}{-}{-}}
\NormalTok{MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCHNLVYaNtGG20xv+mGI9pi9tX/}
\NormalTok{E+Fq0ShuB84X+zfM2sIjss45kHPigmQn47hd8EfCN7DhCe3ynHfPZUyRlMtJAB53}
\NormalTok{qE9rNkV3142chHh+mrJE3MNIp18ooqTLZwGZeZ45c7jV3P2nF/aDlkI7yhnvCC5c}
\NormalTok{2ZGB3kOXyVDBSBWWTwIDAQAB}
\NormalTok{{-}{-}{-}{-}{-}END PUBLIC KEY{-}{-}{-}{-}{-} }
```

KeySize: *1024 Bit* **Signature Algorithm:** *SHA256withRSA*

Nun brauchen Sie noch das Mittel, um die Signatur zu prüfen. Zwar können Mail-Programme Signaturen prüfen, doch dieses Mal nutzen Sie eine Webseite, die das Technische mehr verdeutlicht. Die Webseite wo Sie das Ganze überprüfen können: https://8gwifi.org/rsasignverifyfunctions.jsp

Auf der Seite den die Angaben verwenden:

- Einmal durchführen für Nachricht A und einmal für Nachricht B:
- Signature Algorithm: SHA256withRSA
- Key Size: 1024
- Public Key: Den oben angegebenen Public Key
- Signature: Die oben angegebene Signature
- “Verify” anwenden

Welche Nachricht stimmt mit der Signatur überein und ist somit echt?

Aufgabe: Selber eine Signatur berechnen

Tauschen Sie eine signierte Nachricht mit einem Lernpartner aus.

Die Webseite wo Sie die Signatur erstellen und überprüfen können: https://8gwifi.org/rsasignverifyfunctions.jsp

1. Generieren Sie neue Keys
2. Schreiben Sie eine Message
3. Berechnen Sie die Signatur.
4. Senden Sie die Nachricht, die Signatur und Ihren Public Key an Ihren Lernpartner.
5. Ihr Lernpartner überprüft die Signatur mit Hilfe der Webseite der Nachricht.

Hat es funktioniert? ;-)

Email signieren und verifizieren

Obwohl Signieren eigentlich ein sinnvoller Vorgang ist, wird es in der Praxis wenig angewandt. Der Einsatz bei Emails ist gutes Beispiel. Doch Achtung: Sucht man nach Anleitunge, dann findet man oft die Anleitung für eine Signatur wie zBsp:

```
\NormalTok{{-}{-}{-}{-}}
\NormalTok{Pionierpark Gmbh}
\NormalTok{Peter Rutschmann}
\NormalTok{Mobile: 079 123 45 67}
\NormalTok{{-}{-}{-}}
```

Das ist aber nicht die Art von Signatur, die hier gemeint ist. Es geht um eine digitale Signatur.

Es gibt verschiedene Standards, um Emails digital zu signieren. Die bekanntesten sind PGP (Pretty Good Privacy) und S/MIME (Secure/Multipurpose Internet Mail Extensions).

- PGP ist ein Verschlüsselungsprogramm, das sowohl für die Verschlüsselung als auch für die digitale Signatur von Emails verwendet wird. Mehr zu PGP: https://de.wikipedia.org/wiki/Pretty_Good_Privacy
- S/MIME ist ein Standard für die Verschlüsselung und digitale Signatur von Emails, der auf Zertifikaten basiert. Mehr zu S/MIME: https://de.wikipedia.org/wiki/S/MIME

Weiter hängt es davon ab, welches Email-Programm verwendet wird. Beispiele: Outlook, Thunderbird, Gmail, Apple Mail, etc. Anleitungen findet man in der Dokumentationen des jeweiligen Programmes.

Aufgabe EMail signieren mit GMail

Damit wir gemeinsam das Signieren und Verifizieren von EMails üben können und nicht abhängig von individuellen EMail-Progammen der Lernenden sind, nutzen wir GMail. Ein Vorteil ist auch, dass Sie keine bestehende EMail-Adresse verwenden müssen. Sie kommen sehr einfach zu einer neuen Gmail-Adresse. Weiter können Sie **über das Webinterface von GMail** arbeiten, ohne ein EMail-Programm installieren zu müssen. Allerdings **brauchen Sie ein Browserplugin**, das das Signieren und Verifizieren von Emails ermöglicht.

Schritte:

1. Entscheiden Sie, ob Sie eine bestehende GMail-Adresse verwenden wollen, oder ob Sie eine neue GMail-Adresse erstellen wollen.
2. Stellen Sie sicher, dass die **GMail-Adresse funktioniert**. Senden Sie sich eine EMail an Ihre GMail-Adresse und kontrollieren Sie, ob Sie die Nachricht empfangen. (Noch ohne Signatur)

---

3. Nun folgen Sie der Anleitung für das **Installieren des Browserplugins** von flowcrypt: https://flowcrypt.com/docs/getting-started/setup/install.htmlFolgen Sie der Anleitung auch in Bezug auf das Erstellen eines Schlüsselpaares.
   - Das Plugin muss Zugriff auf Ihr GMail-Konto erhalten.
   - Und Sie müssen ein Key-Paar erstellen (public und private key).

---

4. ACHTUNG: flowcrypt hat zwei unterschiedliche Möglichkeiten
   - EMail verschlüsseln
   - EMail signieren –> **das wollen wir!!**
5. Folgen Sie der Anleitung für das Signieren. https://flowcrypt.com/docs/getting-started/send-and-receive/send/send-signed-only-emails.html Senden Sie eine signierte EMail an Ihre eigene GMail-Adresse. Kontrollieren Sie, ob Sie die signierte Nachricht empfangen.

---

6. Nun senden Sie eine EMail-Nachricht an Ihren Lernpartner, an dessen Gmail-Adresse.Kann Ihr Lernpartner die Echtheit der Nachricht bestätigen?

Zusatz-Aufgabe:EMail signieren mit anderem EMail-Programm

Wahrscheinlich verwenden Sie normalerweise ein anderes Email-Programm und andere Email-Adressen. Informieren Sie sich, ob und wie Sie mir Ihrem normalen Email-Programm Emails signieren und verifizieren können. Richten Sie die das Signieren und Verifizieren ein und testen Sie es mit einem Lernpartner.
