

Kryptographie

Laborprotokoll Übung 3



Figure 1: Wunderbares Gruppenbild

Unterrichtsgegenstand: ITSI|ZIVK
Jahrgang: 3AHITN
Name: Stefan Fürst, Marcel Raichle
Gruppenname/Nummer: Dumm und Dümmer/7
Betreuer: ZIVK
Übungsdaten: 4.10.2024, 11.10.2024
Abgabedatum: 7.6.2024

Contents

1	Aufgabenstellung	3
2	Zusammenfassung	3
3	Übungsdurchführung	4
3.1	Symmetrisch Verschlüsseln	4
3.1.1	Passwort für die Symetrische verschlüsselung berechnen	4
3.1.2	Datei Symetrisch mit AES256 verschlüsseln	4
3.2	Asymmetrisch Verschlüsseln	5
3.3	Integrität prüfen	5
4	Quellen	6
5	Abbildungsverzeichnis	7

1 Aufgabenstellung

Zunächst geht es um die symmetrische Verschlüsselung, bei der eine Datei mit einem berechneten Passwort verschlüsselt und anschließend wieder entschlüsselt wird. Hierbei dient dasselbe Passwort sowohl zur Verschlüsselung als auch zur Entschlüsselung, um den Prozess zu überprüfen und zu verifizieren.

Im zweiten Teil wird die asymmetrische Verschlüsselung behandelt. Dabei werden ein privater und ein öffentlicher Schlüssel generiert, und die Datei wird mithilfe des öffentlichen Schlüssels verschlüsselt. Dieser Ansatz simuliert ein typisches Verschlüsselungsverfahren, bei dem der private Schlüssel zur Entschlüsselung verwendet wird.

Zum Abschluss erfolgt eine Integritätsprüfung mit Hilfe von Hashwerten. Dabei werden mehrere Textdateien mit vorgegebenen Hashwerten verglichen, um sicherzustellen, dass keine Datenveränderungen stattgefunden haben. Ziel ist es zudem, einen Hashwert zu identifizieren, der keiner der Textdateien zugeordnet werden kann.

2 Zusammenfassung

Bei dieser Übung wurden zwei Dateien mit zwei verschiedenen Verfahren verschlüsselt werden, welche Symmetrisch mit AES und Asymmetrisch mit RSA sind. Hierfür wurde das kryptographische Toolkit openssl benutzt.

3 Übungsdurchführung

3.1 Symmetrisch Verschlüsseln

3.1.1 Passwort für die Symmetrische Verschlüsselung berechnen

$$\frac{\text{Datum} + \text{Katalognummer}}{2}$$
$$\frac{20241004 + 24}{2}$$

3.1.2 Datei Symmetrisch mit AES256 verschlüsseln

Hierfür benutzt man openssl, ein kryptographisches Toolkit.[1]

Um in diesem Fall die Datei mit AES256 zu verschlüsseln, man verwendet aes256 als Argument und die Flags -in/out geben die input/outputdatei an. Nach der Eingabe des Commands, muss man ein Passwort eingeben.

```
aufgaben/aufgabe1/datein_zum_arbeiten on ↵ main [!?]
> openssl aes256 -in Raichle-Fuerst.txt -out Raichle-Fuerst-RSA.txt.zivk.enc
enter AES-256-CBC encryption password:
```

Figure 2: AES Verschlüsselung

Für die Entschlüsselung wird die -d Flag verwendet, welche für decrypt steht, dies und das Austauschen von input und output werden benötigt, um die Datei zu entschlüsseln. Wenn der Command ausgeführt wurde, wird das Passwort abgefragt.

```
aufgaben/aufgabe1/datein_zum_arbeiten on ↵ main [!]
> openssl aes256 -d -in Raichle-Fuerst.encrypted -out Raichle-Fuerst.txt
enter AES-256-CBC decryption password:
```

Figure 3: AES Entschlüsselung

```
#verschlüsseln
openssl aes256 -in Raichle.txt -out Raichle.encrypted
#entschlüsseln
openssl aes256 -d -in Raichle.encrypted -out Raichle.txt
```

3.2 Asymmetrisch Verschlüsseln

Für die Asymmetrische Verschlüsselung, müssen erst ein Keypair generiert werden. Für dies werden zwei Commands benötigt, einer für den privaten und einer für den public key.


Bei der Erstellung des public keys, werden der Algorithmus, Schlüsselbits und Dateiname angegeben.

Für die Verschlüsselung wird die `-encrypt` Flag, neben anderen Flags für Key und Dateiinput verwendet um die Datei zu verschlüsseln.

```
#public key
openssl genpkey -algorithm RSA \
-pkeyopt rsa_keygen_bits:4096 \
-out private-key.pem
#private key
openssl pkey -in private-key.pem -out public-key.pem -pubout
#datei verschlüsseln
openssl rsautl -encrypt \
-inkey zivk.pem \
-pubin -in Raichle-Fuerst-RSA.txt \
-out Raichle-Fuerst-RSA.txt.zivk.enc
```

3.3 Integrität prüfen

```
#benötigter command
sha256sum <dateiname>
```



```
1 591ad652f7332fdca28e4ecc520ad7b71852cdad7ab3efbaeeb6042a815c812d
1 e05c11789a98a495d7283a499b1ccc31c368d3c191a4fb5b7074161c816da2e3
2 95697ff6295bc5383b48b8e798348f7ba973adae40090c6c1fac2a5238ea0066
3 e605bd1f525b133340d704f0e899d977f37dea63c14b243a346f1b524499bcf5
4 a20fb601802f7b87b2063964e0d2f7e15b2448bc6ee64dd7a9099991723a2666
```

Figure 4: Hashes

4 Quellen

References

- [1] cheat.sh/openssl, October 2024. [Online; accessed 11. Oct. 2024].

5 **Abbildungsverzeichnis**

List of Figures

1	Wunderbares Gruppenbild	1
2	AES entschlüsselung	4
3	AES entschlüsselung	4
4	Hashes	5