Inhaltsverzeichnis

1. Kryptografie - Begriffe

1.1. Ziele

☑ Grundlagen und Begriffe der IT-Sicherheit kennenlernen.

☑ Quellen:

- ☐ S. Spitz, et.al.(2011): „Kryptographie und IT-Sicherheit“ , Vieweg+Teubner, 2011
- ☐ http://verplant.org/facharbeit/html/node3.html
- ☐ http://kryptologie.kr.funpic.de/grundlagen.html

1.2. Begriffe

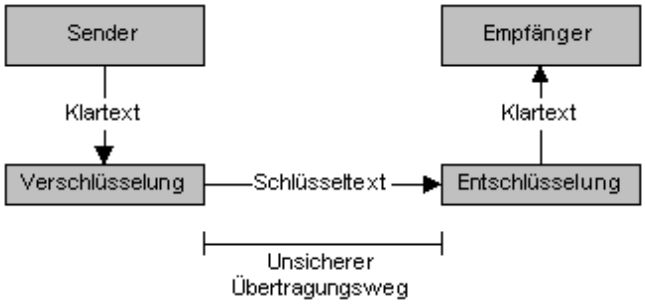
1.2.1. Wissenschaftliche Begriffe

Kryptographie
ist die **Wissenschaft der Verschlüsselung von Informationen.**

Kryptoanalyse
ist die Wissenschaft, kryptographische Algorithmen zu untersuchen. Wird u.a. auch „**Knacken**“ einer Verschlüsselung genannt.

Kryptologoie
ist **Kryptographie + Kryptoanalyse.**

1.2.2. Arten der Verschlüsselung



aus: Handbuch der Java Programmierung (www.javabuch.de , 2012)

Klartext M
(oder „plain text“) ist eine Nachricht, die über einen unsicheren Kanal zu einem Kommunikationspartner übertragen werden soll.

Chiffre-Text C
(oder „cipher text“) ist die verschlüsselte Nachricht.

Symmetrische Verschlüsselung
zum Ver- und Entschlüsseln den **selben Schlüssel** verwenden.
(schnell)

Asymmetrische Verschlüsselung
zum Ver- und Entschlüsseln ein **Schlüsselpaar** (=private key + public key) verwenden.
(langsam)

Monoalphabetische Verschlüsselung
wenn aus einem Klartext-Zeichen **immer dasselbe Chiffre-Zeichen** entsteht.
z.B.: Caesar-Verschlüsselung.
http://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung

Polyalphabetische Verschlüsselung
wenn aus einem Klartext-Zeichen **nicht immer dasselbe Chiffre-Zeichen** entsteht.
z.B.: Vigenere-Verschlüsselung
http://de.wikipedia.org/wiki/Polyalphabetische_Substitution

1.2.3. Schlüssel

Schlüssel
ist die **geheime** Komponente, die beim Ver- und Entschlüsseln angegeben werden muss.
Er ist NUR seinem Eigentümer bekannt.
Er kann z.B. mit dem Programm *ssh-keygen* erzeugt werden.
http://en.wikipedia.org/wiki/Ssh-keygen

private key
ist die geheime Hälfte eines Schlüsselpaares und wird zum

1. **Entschlüsseln** und
2. **Signieren** (unterschreiben) verwendet.

public key

ist der öffentliche Schlüssel eines Schlüsselpaares und wird zum
1. **Verschlüsseln** und
2. **Verifizieren** von Signaturen (Unterschriften) verwendet.

Schlüsselaustauschproblem

wenn ein geheimer Schlüssel über einen unsicheren Kanal ausgetauscht werden soll.

z.B.: Diffie-Hellman Algorithmus (s.u.)
http://de.wikipedia.org/wiki/Diffie-Hellman-Schl%C3%BCsselaustausch

z.B: oder durch Verwendung von private/public keys kann der geheime Schlüssel für die schnellere symmetrische Verschlüsselung ausgetauscht werden. (s. SSL)
Der geheime Schlüssel wird mit dem public-key des Kommunikationspartners verschlüsselt. Nur der Kommunikationspartner kann dann mit seinem private-key den verschlüsselten geheimen Schlüssel entschlüsseln.

Geheimhaltung des Schlüssels

Es muss **nur der Schlüssel geheim** gehalten werden.
Der verwendete Algorithmus kann durchaus bekannt sein.

1.2.4. Zertifikate

☑ **Frage:**
Wer garantiert, dass der PUBLIC-KEY wirklich vom Server-Betreiber ist und nicht vom MAN-IN-THE-MIDDLE?
[*http://en.wikipedia.org/wiki/Man-in-the-middle_attack* 2013.04]

☑ **Antwort:**

☐ **Zertifikte**
(sind von Zertifizierungsstellen unterschriebene PUBLIC-KEYs) und

☐ **Zertifizierungsstellen (CA=Certification Authorities)**

☑ **Signatur**
ist eine „**elektronische Unterschrift**“ und
wird durch verschlüsseln mit dem PRIVATE-KEY erreicht. (= Signieren)

1.2.5. Fragen: Kryptografie – Begriffe

s. Moodle Kurs: KRYPTO/MAB: Krypto – Grundlagen

https://elearning2.htl-salzburg.ac.at/Moodle/moodle/mod/quiz/view.php?id=1184

1.3. Sicherheits-Dienste

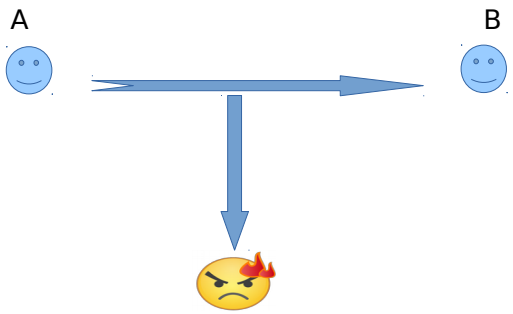
Sicherheitsdienste
sind **Leistungen**, die ein Benutzer in Anspruch nehmen möchte.

Solche Sicherheitsdienste sind:

1. **Vertraulichkeit** – kein Dritter hört zu.
2. **Authentizität** – von wem stammt die Nachricht wirklich?
3. **Integrität** – unverschlüsselte Nachricht wurde nicht manipuliert.
4. **Verbindlichkeit** – der Sender der Nachricht steht eindeutig fest.

1.3.1. Vertraulichkeit - kein Dritter hört zu

„**KEIN Dritter hört zu**“

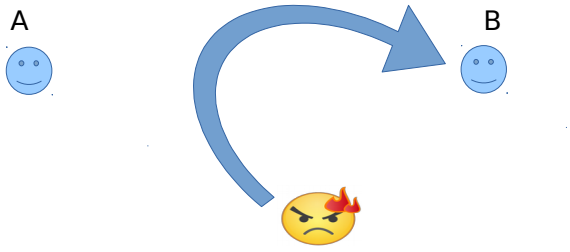


Lösung durch **Verschlüsselung**:

- symmetrischen Schlüssel: „Nur beide kennen ein Geheimnis“
- asymmetrischen Schlüssel: Sender A benutzt den PUBLIC-KEY vom Empfänger B

1.3.2. Authentizität – von Wem stammt die Nachricht wirklich

„Von WEM stammt die Nachricht wirklich?“



Lösung:

- **Sender A signiert die Nachricht durch seinen PRIVATE-KEY.**

Man sagt:

- Sender A **authentisiert** seine Daten
  - durch seinen private-key
- Empfänger B **authentifiziert** die Daten
  - durch den public-key des Sender A.
  - Dabei wird die Echtheit des public-keys durch das Zertifakt einer CA garnatiert. (s. SSL)

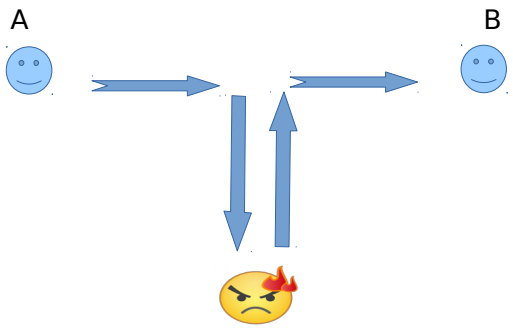
Beispiel:

- Login bei
  - Computer, Bankomat, ...

Diesen Vorgang nennt man **Authentifikation**.

1.3.3. Integrität – unverschlüsselte Nachricht wurde nicht verändert?

„Daten sind vor Veränderung geschützt obwohl sie nicht verschlüsselt wurden“



Lösung: (für Integrität)

1. Sender: übertragen werden
   1. die Nachricht (unverschlüsselt) und
   2. Aus der Nachricht wird ein **Fingerprint** (=Message Digest) erstellt, der wesentlich kleiner als die Nachricht ist.
      SHA: https://de.wikipedia.org/wiki/Secure_Hash_Algorithm
2. Empfänger:
   1. erstellt von der empfangenen Nachricht ebenfalls einen Fingerprint und
   2. vergleicht diesen mit dem erhaltenen Fingerprint.
   3. Bei Gleichheit wurden die Daten nicht verändert.

Hinweis: Hash-Algorithmen https://de.wikipedia.org/wiki/SHA-2

```
$>echo "Hello" |sha256sum
66a045b452102c59d840ec097d59d9467e13a3f34f6494e539ffd32c1bb35f18  -
$>echo "Hallo" |sha256sum
78fca7a0dbd0325b8f77333c82fb1ba2a5cbf9e90284bd24e91cb58ac1d6232f  -
$>sha256sum readme.txt
3849ce34f74ba0e03ffbd771fd8849dbd97453f3c5b1aa22ca8e351c5cf10b20 readme.txt
```

http://releases.ubuntu.com/16.04.1/SHA256SUMS

```
4bcec83ef856c50c6866f3b0f3942e011104b5ecc6d955d1e7061faff86070d4 *ubuntu-16.04-desktop-amd64.iso
b20b956b5f65dff3650b3ef4e758a78a2a87152101a04ea1804f993d8e551ceb *ubuntu-16.04-desktop-i386.iso
b8b172cbdf04f5ff8adc8c2c1b4007ccf66f00fc6a324a6da6eba67de71746f6 *ubuntu-16.04-server-amd64.img
b8b172cbdf04f5ff8adc8c2c1b4007ccf66f00fc6a324a6da6eba67de71746f6 *ubuntu-16.04-server-amd64.iso
8d52f3127f2b7ffa97698913b722e3219187476a9b936055d737f3e00aecd24d *ubuntu-16.04-server-i386.img
8d52f3127f2b7ffa97698913b722e3219187476a9b936055d737f3e00aecd24d *ubuntu-16.04-server-i386.iso
dc7dee086faabc9553d5ff8ff1b490a7f85c379f49de20c076f11fb6ac7c0f34 *ubuntu-16.04.1-desktop-amd64.iso
cea23ae1ce57e7ee2495b74cc232358523d8d0a754a3aa3dd7d8f9d55408f5ae *ubuntu-16.04.1-desktop-i386.iso
29a8b9009509b39d542ecb229787cdf48f05e739a932289de9e9858d7c487c80 *ubuntu-16.04.1-server-amd64.img
29a8b9009509b39d542ecb229787cdf48f05e739a932289de9e9858d7c487c80 *ubuntu-16.04.1-server-amd64.iso
62fc3e810c7631fbbd45d9c960ec749fc1eb66e5c56039423e3e94a5391a437a *ubuntu-16.04.1-server-i386.img
62fc3e810c7631fbbd45d9c960ec749fc1eb66e5c56039423e3e94a5391a437a *ubuntu-16.04.1-server-i386.iso
```

auch openssl (s. SSL) bietet Hash-Funktionen

- To create a hex-encoded message digest(=Finger print) of a file:

  openssl dgst -sha256 -hex file.txt

  ```
  SHA256(file.txt)= e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
  ```

☑ **Problem**:
Verbindlichkeit, d.h. **war die Nachricht wirklich vom Sender?**
Es kann **nicht** eindeutig bestimmt werden, **von wem** die Nachricht wirklich stammt.

Auch der Empfänger B könnte sich eine derartige Nachricht zusammenstellen und behaupten, diese von A erhalten zu haben.

☑ **Lösung:**
Dieses Problem löst der Sicherheits-Dienst namens **Verbindlichkeit**, der auch **Nichtabstreitbarkeit** genannt wird.

1.3.4. Verbindlichkeit (Nichtabstreitbarkeit) – der Sender steht eindeutig fest

„**Der Urheber einer Nachricht ist eindeutig bestimmbar**“

Sogar vor Gericht kann eindeutig bestimmt werden, wer der Urheber einer Nachricht war.

Die Eigenschaft der Verbindlichkeit schließt die Authentizität u. Integrität ein.

Verbindlichkeit := Authentizität + Integrität

Lösung allgemein: Den Fingerprint einer Nachricht signieren

- Durch eine **digitale Signatur** (Unterschrift) kann Verbindlichkeit erreicht werden.
- Hier wird der **PRIVATE-KEY** eines asymmetrischen Schlüsselpaares genutzt.

- Zum PRIVATE-KEY hat ausschließlich sein Besitzer Zugang.

  Nur er kann diese Signatur (Unterschrift) leisten.
- Die Signatur kann von jedem, der den PUBLIC-KEY hat verifiziert werden.

Lösung 1: (für Authentizität u. Integrität mit symmetrischer Signatur)

1. Aus der Nachricht wird ein **Fingerprint** (=Message Digest) erstellt,
   der wesentlich kleiner als die Nachricht ist.
   SHA: https://de.wikipedia.org/wiki/Secure_Hash_Algorithm
2. **Symmetrischer Schlüssel** (der nur A und B bekannt ist) verwenden
3. **Nur Fingerabdruck wird durch symmetrischen Schlüssel** verschlüsselt.
   (vgl.: **MAC = Message Authentication Code**)

Integritätsverletzung liegt vor, wenn:

1. der Empfänger B den Fingerabdruck der Nachricht bildet und
2. den erhaltenen verschlüsselten Fingerabdruck mit dem symmetrischen Schlüssel entschlüsselt.
3. Wenn beide Fingerabdrücke nicht übereinstimmen.

Authentizität ist wegen der symmetrischen Schlüssel gegeben.

Lösung 2: (für Authentizität u. Integrität mit Asymmetrischer Signatur)

1. Aus der Nachricht wird ein **Fingerprint** (=Message Digest) erstellt.
2. **Nur Fingerprint wird durch PRIVATE-KEY** signiert/verschlüsselt.
   (vgl.: **MAC = Message Authentication Code**)

Integritätsverletzung liegt vor, wenn:

4. der Empfänger B den Fingerabdruck der Nachricht bildet und
5. den erhaltenen signierten/verschlüsselten Fingerabdruck mit dem PUBLIC-KEY verifiziert/entschlüsselt.
6. Wenn beide Fingerabdrücke nicht übereinstimmen.

Hinweis: openssl und MAC

1. Private-key (RSA 4096 Bits) erzeugen

```
openssl genrsa  -out private.pem  4096
```

2. Schlüssel Leseschutz

```
chmod 600 private.pem
```

3. Public-Key erzeugen

```
openssl rsa -pubout -in private.pem -out public.pem
```

4. Fingerprint einer Textdatei erzeugen, wenn nur Integrität gefordert

```
openssl dgst -sha512 -out fingerprint_LONGTEXT.txt LONGTEXT.txt
```

5. Fingerprint einer Textdatei erzeugen und diesen Fingerprint signieren, wenn NICHTABSTREITBARKEIT (Integrität und Authentizität) gefordert

```
openssl dgst -sha512 -sign private.pem -out fingerprint_LONGTEXT.txt -hex
LONGTEXT.txt
```

6. Eine Signatur/Unterschrift (MAC) verifizieren

```
openssl dgst -sha256 -verify public.pem \
        -signature fingerprint_LONGTEXT.txt \
        LONGTEXT.txt
```

Hinweis: Den Empfang einer Nachricht signieren.

- Es kann auch der Empfang einer Nachricht durch Signieren des Empfängers quittiert werden.

1.3.5. Aufgabe: AB-MAC

Laden Sie die Datei AB-MAC.zip und prüfen Sie die Integrität und Authentizität der übertragenen Datei.

1.4. Zusammenfassung

Sicherheits-Dienst	Problem	Lösung
Vertraulichkeit	ein Dritter hört zu	Nachricht **verschlüsseln**
Authentizität	ein Dritter sendet Daten	Nachricht **signieren**
Integrität	Unverschlüsselte Daten sind nicht vor Veränderung geschützt	**Fingerabdruck** der Nachricht **sym. Verschlüsseln** (MAC)
Verbindlichkeit	Der Urheber ist nicht eindeutig bestimmbar	Nachricht **signieren**

1.5. Fragen

Zum Signieren einer Nachricht verwendet man den eigenen PUBLIC-KEY.
o wahr
o falsch*

Es darf nicht bekannt sein, welchen Verschlüsselungs-Algorithmus man verwendet.
o wahr
o falsch*

Vertraulichkeit wird durch Verschlüsseln mit dem PRIVATE-KEY erreicht.
o wahr
o falsch*

Vertraulichkeit wird durch sym. Verschlüsseln mit dem gemeinsamen KEY erreicht.
o wahr*
o falsch

Den Sicherheits-Dienst: „Von wem stammt die Nachricht“ nennt man Authentizität.
o wahr*
o falsch

Der Sender authentifiziert sich und der Empfänger authentisiert.
o wahr
o falsch*

Authentizität löst man durch Verschlüsseln mit dem PRIVATE-KEY.
o wahr*
o falsch

Es gitl: Verbindlichkeit := Authentizität + Integrität
o wahr*
o falsch

Sicherheits-Dienst	**Problem**	**Lösung**
Vertraulichkeit	ein Dritter hört zu	
Authentizität	ein Dritter sendet Daten	
Integrität	Unverschlüsselte Daten sind nicht vor Veränderung geschützt	
Verbindlichkeit	Der Urheber ist nicht eindeutig bestimmbar	

Welche Zeile muss in die Spalte Lösung?

1. Nachricht verschlüsseln
2. Nachricht signieren
3. Fingerabdruck der Nachricht sym. Verschlüsseln (MAC)
4. Nachricht signieren