Themenkorb 8

a) Hash Verfahren

Was sind Hash-Funktionen

Mit einer Hash-Funktion kann aus einem Eingabewert wie z.B. einer Datei ein Hashwert erzeugt werden. Moderne Hash-Funktionen sind so ausgelegt, dass es praktisch unmöglich ist zwei Eingabewerte zu finden, die den gleichen Hashwert ergeben.

Hash-Funktionen werden dazu verwendet Dateien oder Nachrichten darauf zu überprüfen ob sie verändert bzw. richtig übertragen wurden, dies nennt man Integritätsprüfung.

Der Hashwert ist sozusagen ein digitaler Fingerabdruck, die kleinste Veränderung an der Datei führt zu einem komplett anderen Hashwert.

Eine Hash-Funktion geht nur in eine Richtung, das heißt man kann aus z.B. einer Datei einen Hashwert erzeugen, nicht aber vom Hashwert auf die ursprüngliche Datei schließen.

Bekannte Hash-Verfahren sind MD5, SHA1, SHA256 und SHA512, wobei MD5 bereits geknackt wurde. Somit sollte der MD5-Hash nur mehr für die Überprüfung ob Dateien richtig übertragen wurden und nicht für Passwörter und Signaturen verwendet werden.

Hash-Funktionen im Vergleich zu Prüfsummen

Hash-Funktionen unterscheiden sich sowohl im Anwendungsgebiet als auch in ihrer Funktionsweise von Prüfsummen. Prüfsummen sollten nur zur Überprüfung verwendet werden, ob Dateien unabsichtlich bei der Übertragung beschädigt worden sind.

Sie sollten nicht für sicherheitskritische Anwendungen verwendet werden, da Prüfsummen rückrechenbar sind. Dateien könnten absichtlich ohne großen Aufwand manipuliert werden, sodass sie trotzdem die gleiche Prüfsumme ergeben. Deshalb sollten bei allen sicherheitskritischen Anwendungen Hash-Funktionen verwendet werden.

Prüfsummen wie z.B. CRC sind üblicherweise kleiner als gängige Hash-Verfahren. So hat die häufig verwendete CRC32 Prüfsumme nur 32 Bit im Vergleich zu 128 bis 512 Bit wie bei Hash-Verfahren. Durch die geringe Größe ergibt sich ein kleinerer Rechenaufwand. Somit sind die Prüfsummen üblicherweise schneller.

Anwendung bei Passwörtern

Passwörter sollten aus Sicherheitsgründen nie im Klartext gespeichert werden. Deshalb ist es gang und gäbe nicht das Passwort, sondern den Hashwert des Passworts zu speichern.

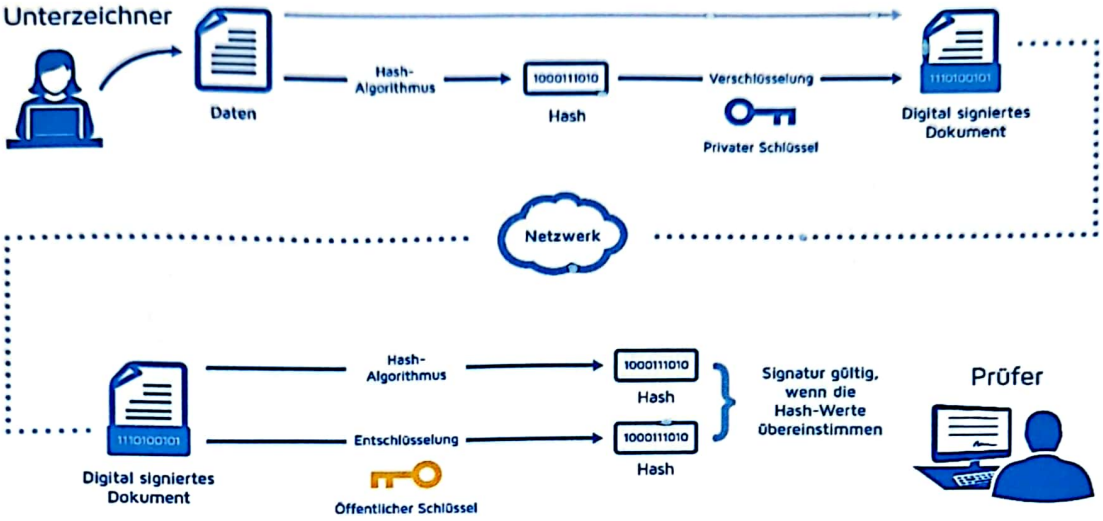
Um Angriffe durch eine Wörterbuchattacke zu erschweren wird bei Hash-Funktionen öfters das sogenannte „Salt" verwendet. Dabei werden beim Klartext des Passworts einige Bits angehängt, sodass sich ein komplett anderer Hash bildet und die Wörterbuchtabellen nutzlos werden. Die Salts werden gemeinsam mit den Hashwerten in der Datenbank gespeichert.

Eine ähnliche Variante Angriffe zu erschweren ist „Pepper". Der Unterschied zum Salt ist, dass der Pepper meist für alle Passwörter gleich ist. Dieser wird an einem möglichst sicheren Ort und NICHT in derselben Datenbank wie die Hashwerte gespeichert.

Anwendung bei Signaturen

Um Daten mit einer digitalen Signatur zu versehen, wird zuerst der Hashwert der Datei errechnet werden. Der Hashwert wird anschließend mit dem Privaten Schlüssel verschlüsselt.

Die Datei inklusive verschlüsseltem Hash kann nun an den Empfänger versendet werden. Dieser muss den Hashwert der Datei errechnen und den verschlüsselten Hash mit dem öffentlichen Schlüssel entschlüsseln. Stimmen die beiden Hashwerte überein, sind die Daten nicht verändert worden.



Beispielprogramm 1

Interpretation der Angabe:
Klasse entwickeln welche aus Collection<Integer> mit einer zusätzlichen Methode den Hash bildet.
Eine weitere Methode soll einen Hash mit dem berechneten Hash vergleichen.

```
public class Hasher {

   private final Collection<Integer> list;
   private final MessageDigest md;
   private byte[] md5;

   public Hasher(final Collection<Integer> list) throws Exception {
      this.list = list;
      md = MessageDigest.getInstance("md5");
   }

   public void calcHash() {
      final byte[] data = list.toString().getBytes();
      md5 = md.digest(data);
   }

   public boolean verify(byte[] hash) {
      return hash.equals(md5);
   }

   public byte[] getMd5() {
      return md5;
   }
}
```

Beispielprogramm 2

Übungsbeispiel aus dem Unterricht

```
public class MD5Calculator {
  private final byte[] md5;

  public MD5Calculator(final byte[] data) throws NoSuchAlgorithmException {
    final MessageDigest md = MessageDigest.getInstance("md5");
    md5 = md.digest(data);
  }

  public MD5Calculator(final String text) throws UnsupportedEncodingException, NoSuchAlgorithmException {
    this(text.getBytes("utf8"));
  }

  private final static int BUFSIZE = 64*1024;

  public MD5Calculator(final File file) throws Exception {
    final MessageDigest md = MessageDigest.getInstance("md5");
    long toRead = file.length();
    final byte[] buffer = new byte[BUFSIZE];

    try(final FileInputStream fis = new FileInputStream(file)) {
      while(toRead > 0L) {
        final int result =
          fis.read(buffer, 0, toRead>=BUFSIZE ? BUFSIZE : (int)toRead);
        if (result == -1)
          break;
        md.update(buffer, 0, result);
        toRead -= result;
      }
    }
    md5 = md.digest();
  }

  public byte[] getMd5() {
    return md5;
  }

  public String getMd5AsString() {
    //return new BigInteger(1,md5).toString(16).toUpperCase();
    return String.format("%032X", new BigInteger(1,md5));
  }
```

b) Ver- und Entschlüsselung

Verschlüsselung

Bei einer Verschlüsselung einer Datei bleiben die Daten lesbar, es kann nur mehr der Informationsgehalt nicht entnommen werden. Beim Entschlüsseln wird der Informationsgehalt entnehmbar gemacht.

Symmetrische und asymmetrische Verschlüsselung

Bei einer symmetrischen Verschlüsselung gibt es nur einen Schlüssel. Mit diesem kann sowohl verschlüsselt als auch entschlüsselt werden.

Bei einer asymmetrischen Verschlüsselung gibt es einen öffentlichen und einen privaten Schlüssel, zusammen ergeben sie ein Schlüsselpaar. Mit dem öffentlichen Schlüssel können die Daten verschlüsselt, aber nicht entschlüsselt werden. Das entschlüsseln ist nur mit dem privaten Schlüssel möglich. Es ist nicht möglich vom öffentlichen Schlüssel den privaten Schlüssel zu errechnen, umgekehrt aber schon.

Eine Asymmetrische Verschlüsselung ist deutlich langsamer als eine symmetrische. Deshalb gibt es die Variante der hybriden Verschlüsselung. Bei ihr wird der symmetrische Schlüssel mit dem die Daten anschließend verschlüsselt werden über eine asymmetrische Verschlüsselung ausgetauscht.

AES (Advanced Encryption Standard)

AES ist ein symmetrisches Verschlüsselungsverfahren mit Schlüssellängen zwischen 128 und 256 Bit. Das Verfahren ist frei verfügbar und darf ohne Lizenzgebühren eingesetzt werden.

RSA

RSA ist ein asymmetrisches Verschlüsselungsverfahren. Es kann sowohl für Verschlüsselungen als auch für digitale Signaturen eingesetzt werden.

Beispielprogramm 2
Übungsbeispiel aus dem Unterricht

```
public class Crypter {

  public static byte[] encode(String text, String pass) throws Exception {
    //Schritt 1: Text in Bytes umwandeln
    final byte[] bText = text.getBytes("utf8");
    //Schritt 2: Passwort in Bytefeld umwandeln
    final byte[] bPass = pass.getBytes("utf8");
    //Schritt 3: MD5 des Passworts berechnen
    final byte[] bKey = MessageDigest.getInstance("md5").digest(bPass);
    //Schritt 4: Schlüsselobjekt erzeugen
    final SecretKeySpec key = new SecretKeySpec(bKey, "aes");
    //Schritt 5: Objekt für Verschlüsselungsalgorithmus erzeugen
    final Cipher cipher = Cipher.getInstance("aes");
    //Schritt 6: Algorithmus initalisieren
    cipher.init(Cipher.ENCRYPT_MODE, key);
    //Schritt 7: Verschlüsseln
    return cipher.doFinal(bText);
  }

  public static String bytesToString(byte[] data) {
    //return new BigInteger(1,md5).toString(16).toUpperCase();
    return String.format(
        //"%032X",
        String.format("%%0%dX", data.length*2),
        new BigInteger(1,data));
  }

  public static String decode(byte[] data, String pass) throws Exception {
    final byte[]
        bPass = pass.getBytes("utf8"),
        bKey = MessageDigest.getInstance("md5").digest(bPass);
    final SecretKeySpec key = new SecretKeySpec(bKey, "aes");
    final Cipher cipher = Cipher.getInstance("aes");
    cipher.init(Cipher.DECRYPT_MODE, key);
    final byte[] encryptData = cipher.doFinal(data);
    return new String(encryptData,"utf8");
  }
}
```

c) REST-Server

OSI-Modell

Im Osi-Modell ist die Datenübertragung im Netzwerk in Schichten unterteilt dargestellt.

#	ISO/OSI	DOD / TCP	Datentyp	Beispiel
7	Anwendung	Anwendung	Daten	HTTP, SMTP,
6	Darstellung			RTP
5	Sitzung			
4	Transport	Transport	Segment	TCP, UDP
3	Vermittlung	Internet	Paket	IP, IPv4
2	Sicherung	Netzzugriff	Frame	Ethernet
1	Bitübertragung		Bit / Symbol	WLAN

Ethernet/Wlan

Ethernet ist eine Technik für kabelgebundene Netzwerke, es werden sowohl Hardware (z.B. Verteiler, Kabel) und auch Software (Protokolle) spezifiziert. Es repräsentiert im OSI-Modell die ersten zwei Schichten. Bei einer Übertragung über Funk verwendet man WLAN, welches sich auf den gleichen Schichten des OSI-Modells befindet. Für die Identifikation der einzelnen Teilnehmer werden die MAC-Adressen verwendet welche statisch auf der Netzwerkkarte gespeichert sind und im Normalfall nicht geändert werden sollten.

In üblichen Netzwerken wird eine Stern-Topologie verwendet, bei ihr gibt es einen zentralen Verteiler (Switch oder Hub) welcher direkt mit jedem anderen Teilnehmer verbunden ist. Die einzelnen Teilnehmer schicken ihre Daten an den Verteiler und dieser schickt diese an die gewünschten Teilnehmer weiter.

Ein Router dient allgemein dazu mehrere Netzwerke miteinander zu verbinden und Daten weiterzuleiten. Eine weitere Funktion, welche die meisten Router besitzen ist NAT (Network Adress Translation), dabei wird zwischen der privaten IP-Adresse im Subnetz und der öffentlichen IP-Adresse übersetzt. Somit ist es möglich mit einem Rechner im Subnetz, auf z.B. einen Server in einem anderen Subnetz zuzugreifen. Üblicherweise regelt der Router auch die IP-Vergabe im Subnetz. Weiters verfügen die meisten Router über eine Firewall, mit welcher nicht benötigte Ports blockiert werden können.

Ein weiterer Begriff dieser Thematik ist der Access Point, welcher beim WLAN vorkommt. Über den Access-Point kann sich ein Host mit dem Netzwerk verbinden, bekommt aber keine IP-Adresse vom Access-Point sondern von dem Router des Netzwerks.

TCP/IP

TCP/IP ist eine Gruppe von Netzwerkprotokollen bestehend aus TCP, IP, UDP und ICMP. Im weiteren Sinne wird die ganze Internet-Protokollfamilie als TCP/IP bezeichnet.

Das IP-Protokoll wird für die Identifikation der einzelnen Teilnehmer, auch Hosts genannt, benötigt. Es gibt IPv4 und IPv6 Adressen. IPv4-Adressen bestehen aus 32 Bits, dadurch ergibt sich eine Anzahl von 2^{32} Adressen. Durch den rasch steigenden Bedarf an IP-Adressen ist absehbar, dass der nutzbare Adressraum von IPv4 früher oder später erschöpft sein wird. Vor allem aus diesem Grund wurde IPv6 entwickelt.

Das TCP-Protokoll sorgt dafür das die Datenpakete richtig übertragen werden und dass diese, wenn Fehler auftreten neu gesendet werden. Weiters steuert das TCP-Protokoll den Verbindungsaufbau und -abbau.

Http (Hypertext Transfer Protokoll)

Http ist ein zustandsloses Protokoll zur Datenübertragung von Maschine zu Maschine. Es wird hauptsächlich dazu verwendet Daten oder Webseiten in den Webbrowser zu laden.

Die wichtigsten Funktionen von http sind in der folgenden Tabelle beschrieben:

GET	Listen/Daten anfragen
POST	Daten verändern
PUT	Neue Daten anlegen
DELETE	Daten löschen

Unterschiede Webprogrammierung und Desktopprogrammierung

Für Web und Desktopprogrammierung verwendet man meist verschiedene Sprachen. So wird im Webbereich meist Javascript verwendet und im Desktopbereich Java. Natürlich können mit Javascript auch Desktopanwendungen erstellt werden, aber dies bringt keine wirklichen Vorteile.

Die beiden genannten Sprachen haben bis auf den Namen aber keine wirklichen Gemeinsamkeiten, außer, dass sie Objektorientiert sind. Sie Unterscheiden sich im Syntax, im Typsystem, bei der Laufzeitumgebung, etc.

Java ist Typenstreng, JavaScript nicht. Typenstreng bedeutet, dass eine Variable einen Datentyp hat, wie z.B. „String" und dieser Variable somit nur ein String als Wert zugewiesen werden kann. Bei einer nicht typenstrengen Sprache wird kein Datentyp angegeben und es kann jeder beliebige Wert oder jedes Objekt dieser Variable zugewiesen werden. Der Vorteil einer nicht typenstrengen Sprache ist eine größere Flexibilität, aber genau diese erweist sich in den meisten Augen als Problem, da es dem Programmierer leicht gemacht wird unsaubere Lösungen (Pfusch) zu finden. Diese unsauberen Lösungen können zwar bei kleinen Programmen praktisch sein, doch machen sie den Programmieren bei größeren Projekten, aufgrund der vielen Fehlermöglichkeiten das Leben schwer. Um in der Webprogrammierung nicht auf Typen verzichten zu müssen wurde die Sprache TypeScript entwickelt. TypeScript ist ein typenstrenges Javascript, welches von einem sogenannten Transpiler in Javascript übersetzt wird. Es ist somit nur eine Hilfestellung für den Programmierer und es kann mit der gleichen Laufzeitumgebung ausgeführt werden, für den User ändert sich somit nichts.

Ein weiterer Unterschied zwischen Java und JS ist die Laufzeitumgebung und der Compiler. Bei Java wird der geschriebene Code mit dem Compiler in einen Maschinencode übersetzt (dem Java Native Code). Der Maschinencode ist aber nicht für den ausführenden Computer, sondern für die JVM (Java Virtual Machine). Beim kompilieren wird der Code auf Fehler überprüft und lässt sich nur kompilieren, wenn er fehlerfrei ist. Bei Javascript hingegen gibt es die sogenannte Just-in-time-Compilation. Dabei wird der Quellcode erst während dem ausführen übersetzt. Somit können Fehler erst während der Laufzeit erkannt werden. Das erschwert das debuggen enorm, wäre anders aber schwer möglich, da Webseiten sonst sehr lange Ladezeiten hätten. Als Laufzeitumgebung dient bei Javascript Serverseitig meist NodeJS und Clientseitig der Browser. Sowohl NodeJS als auch die meisten Browser basieren auf der Chrome V8 Engine.

Beispielprogramm

Was ist NodeJS? NodeJS ist eine Plattform für den Betrieb von Netzwerkanwendungen wie z.B. Webservern. Es basiert auf der V8 Engine, welche ursprünglich für Chrome entwickelt wurde. Node lässt sich einfach mit dem Node Package Manager um Module erweitern. Diese Module enthalten Tools, Frameworks, etc. Beispiele für Module wären z.B. Express, Body-Parser oder Angular.

Bsp. Hinzufügen eines Moduls: `npm install--save express`

Wenn statt JavaScript mit TypeScript programmiert wird müssen auch die Typen für das jeweilige Modul hinzugefügt werden.

Bsp. Hinzufügen der Typen: `npm install --save-dev @types/express`

Was ist Express? Express ist ein Framework für die Realisierung von HTTP Server. Es kann mit dem Node Package Manager installiert werden.

Bsp. Implementierung von Express in Serverklasse

```
// Externes Modul
import * as express from 'express';

export class Server {
  private _port: number;
  private _server: express.Express;

constructor (port: number) {
    this._port = port;
    this._server = express();
    this._server.get('/liste', (req, res, next) => this.handleGetListe(req, res, next));
    this._server.get('/image.png', (req, res, next) => this.sendImage(res));
  }
  public start () {
    this._server.listen(this._port);
    console.log('HTTP Server gestartet auf port ' + this._port);
  }

  public get port () {
    return this._port;
}

private handleGetListe(req: express.Request, res: express.Response, next: express.NextFunction) {  // any = irgend ein datentyp
    // res.send('Guten Morgen, Herr Muri');
    const filePath = path.join(__dirname, '..', 'assets', 'liste.html');
    // damit wird der Dateipfad zusammengefügt/notwendig damit unsere Datei gefunden wird
    console.log(filePath);
    res.sendFile(filePath);
    // res.end(); // ende der Anfrage (Response geschickt)
    // res.end() beendet die response zu schnell das die Datei nicht mal übertragen werden kann
  }
  // next ist dafür, da falls die Anfrage nicht sofort bearbeitet werden kann
  // und gibt sie der nächsten schicht weiter
}
```

Starten des Servers in der Mainklasse

```
import { Server } from './server';


class Main {

  public static main () {
    const server = new Server(4711);
    server.start();
  }
}

Main.main();
```

Sicherheitsrisiken

Bei der Lösung mit HTTP ergeben sich durch die unverschlüsselte Übertragung Sicherheitsrisiken, da theoretisch jeder die Kommunikation im Klartext mitlesen kann. Dadurch fällt es Hackern leicht Man-in-the-Middle-Angriffe durchzuführen. Die Lösung für das Problem ist die Verwendung von HTTPS. Bei HTTPS geschieht eine Verschlüsselung der Daten mit SSL/TLS. Dabei wird das SSL-Handshake-Protokoll zur Authentifizierung des Kommunikationspartners verwendet. Anschließend wird mit asymmetrischer Verschlüsselung ein symmetrischer Sitzungsschlüssel ausgetauscht mit welchem die Nutzdaten anschließend verschlüsselt werden.

d) Sicherer Webserver tlsx509, REST-Server mit verschlüsselter Übertragung

HTTPS (Hypertext Transfer Protokoll Secure)

HTTPS ist http mit integrierter TLS/SSL Verschlüsselung. Das Protkoll wurde von Netscape entwickelt und dient zur Herstellung von Vertrauen und Integrität bei der Kommunikation zwischen Webserver und Webbrowser. Dies wird durch Verschlüsselung und Authentifizierung erreicht. Für die Authentifizierung werden Zertifikate auf Basis des X509 Standards verwendet.

TLS/SSL

SSL (secure Socket Layer) ist die alte Bezeichnung für TLS (Transport Layer Security). Es handelt sich dabei um ein hybrides Verschlüsselungsprotokoll, das heißt, dass der symmetrische Schlüssel für das verschlüsseln der Daten zuerst über ein asymmetrisches Verschlüsselungsverfahren ausgetauscht wird. Dabei wird häufig AES als symmetrisches und RSA als asymmetrisches Verfahren verwendet

X509

Bei X509 handelt es sich um eine sogenannte „Public-Key-Infrastruktur" zum Erstellen digitaler Zertifikate. Einfacher gesagt ist es ein Standard, nach welchem digitale Zertifikate ausgestellt werden können. X509 Zertifikate kommen häufig bei TLS zum Einsatz und werden dazu benötigt die Echtheit des öffentlichen Schlüssels zu gewährleisten. Ohne eines solchen Zertifikats wäre es nicht möglich festzustellen ob der öffentliche Schlüssel des Verbindungspartners nicht durch eine Man-in-the-Middle-Attacke korrumpiert wurde. Der X509 Standard sieht vor, dass digitale Zertifikate nur von vertraulichen Zertifizierungsstellen ausgegeben werden können. Der Webbrowser verfügt über eine Liste von Zertifizierungsstellen, denen der Browser vertraut. Ein digitales Zertifikat kann auch selbst erstellt werden, nur wird dieses vom Browser nicht standardmäßig vertraut. Man spricht dann von einem selbstsignierten Zertifikat.

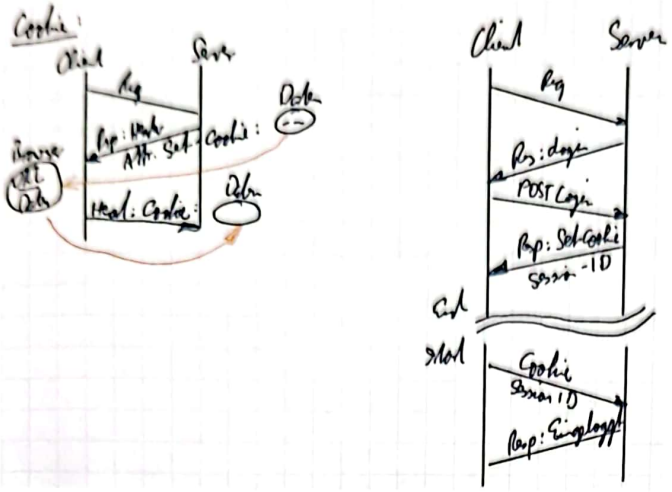
Eine Möglichkeit ein solches zu erstellen ist OpenSSL.

Dabei wird zuerst eine CA (Certificate Authority) erstellt. Diese ist im Grunde ein RSA Schlüsselpaar. Danach wird das Root-Zertifikat erstellt, dieses besteht aus Infos über den Aussteller und dem Öffentlichen Schlüssel der CA. Dieses Root-Zertifikat muss nun auf die Clients importiert werden. Mithilfe der CA können nun beliebig viele Zertifikate für verschiedene Server erstellt werden. Die Clients benötigen nur das Root-Zertifikat um eine gesicherte Verbindung mit den verschiedenen Servern aufzubauen.

Cookies

Ein Cookie ist eine Textinformation, die der Server über den Webbrowser am Client platziert. Ein Cookie wird entweder vom Webserver im Header mitgesendet oder durch ein Skript (z.B. mit JavaScript) direkt am Browser erzeugt. Durch Anforderung des Servers sendet der Client die Cookie-Informationen wieder an den Server. Verwendungszwecke für Cookies sind z.B. Abspeichern eines Logins oder das tracken der Benutzer. Oft zur Anwendung kommt auch eine SessionID, welche dazu dient den Client bei mehreren Aufrufen wiederzuerkennen. Um mit allen Browsern kompatibel zu bleiben dürfen Cookies eine maximale Größe von 4kB aufweisen. Cookies können wahlweise vom Benutzer im den Browser-einstellungen aktiviert werden. Dadurch funktionieren manche Webseiten nicht wie es von den Betreibern vorgesehen ist, schützt aber gegen Tracking. Beim Tracking wird das Surfverhalten des Benutzers gespeichert um gezielt Werbung machen zu können. (Damit verdienen Facebook, usw. ihr Geld)

Deshalb wurde im neuen Datenschutzgesetz vorgeschrieben, dass Webseiten den Benutzer fragen müssen ob dieser Cookies erlaubt oder nicht.



JSON Webtoken

Eine modernere Alternative zu Cookies ist das JSON-Webtoken. Das JWT ist ein auf JSON basierendes Access Token und besteht aus dem Header, der Payload (Inhalt) und der Signatur. Übertragen wird das JWT über HTTP. Das JWT ermöglicht weiters eine sogenannte „stateless session", bei welcher der Server die Sitzung nicht speichern muss, da die Daten alle in einem JWT am Client gespeichert werden.

Beispielprogramm

Um einen HTTP-Server auf HTTPS umzustellen muss zuallererst ein X509 Zertifikat für den Server erstellt werden und auf den Server übertragen werden. Anschließend muss das Programm angepasst werden.

In der Server Klasse muss folgender Import getätigt werden:

```
import * as https from 'https';
```

Weiters muss ein credentials Objekt mit dem Pfad des Schlüssels und des Zertifikats erstellt werden:

```
const credentials = {key: '/home/pi/.ssh/id_rsa', cert: '/home/pi/ue05/certificate/crt.crt'};
```

Als letzter Schritt muss beim erstellen des Servers das credentials Objekt mit übergeben werden:

```
const server = https.createServer(credentials, this._express).listen(this._config.port, () => {
    debug.info('Server gestartet: http://localhost:%s', this._config.port);
});
```

Das restliche Programm kann ohne Änderungen übernommen werden.