FOM Hochschule für Oekonomie & Management

Hochschulzentrum Münster

Seminararbeit

im Studiengang Wirtschaftsinformatik

im Rahmen der Lehrveranstaltung
Fallstudie / Wissenschaftliches Arbeiten

über das Thema

Kryptografie: Verwendung von mathematischen Methoden wie Verschlüsselung und digitalen Signaturen zur Sicherung von Webanwendungen

von

Joshua-Volkan Gramatzki

Betreuer:	Prof. Dr. Gregor Hülsken
Matrikelnummer	647100
Abgabedatum	21. Juni 2023

Inhaltsverzeichnis

Abbildungsverzeichnis **iv**

Tabellenverzeichnis **v**

Abkürzungsverzeichnis **vi**

1 Einleitung **1**
1.1 Zielsetzung 1
1.2 Aufbau der Arbeit 1

2 Grundlagen der Kryptografie - Kryptografische Verfahren und Algorithmen **2**
2.1 Symmetrische Verschlüsselungsalgorithmen 2
2.2 Asymmetrische Algorithmen 3
2.2.1 RSA-Verfahren 3
2.2.1.1 Schlüsselerzeugung 3
2.2.1.2 Verschlüsselung 4
2.2.2 Hashfunktionen 4

3 Anwendung von Kryptografie in Webanwendungen **6**
3.1 Sicherheitsanforderungen an Webanwendungen 6
3.2 Verschlüsselung von Datenübertragungen 7
3.2.1 Digitale Signatur - SSL/TLS-Verschlüsselung 7
3.2.1.1 Digitale Signatur 7
3.2.1.2 SSL/TLS-Verschlüsselung 7
3.2.2 HTTPS-Protocol 9
3.3 Passwortsicherheit 9
3.3.1 Schlüsselableitung 9
3.3.2 Salted Hashing 9
3.3.3 Peppered Hashing 10
3.4 Authentifizierung und Autorisierung 10
3.4.1 Access Token 11
3.4.2 OAuth 12
3.4.3 2-Faktor-Authentifizierung 14

4 Herausforderungen bei der Implementierung von kryptografischen Funktionen in Webanwendungen **15**
4.1 Performance- und Skalierbarkeitsprobleme 15

4.2 Komplexität und Fehleranfälligkeit der Kryptografischen Methoden 17
4.2.1 Komplexität bei der Implementierung von kryptografischen Methoden 17
4.2.2 Fehleranfälligkeit von kryptografischen Methoden in Webanwendungen 18
4.3 Benutzerfreundlichkeit und Usability-Aspekte 18

5 Fazit **20**

Literaturverzeichnis **21**

Abbildungsverzeichnis

Abbildung 1: Schaubild einer symmetrischen Verschlüsselung 2
Abbildung 2: Darstellung eines SSL/TLS-Handshakes 8
Abbildung 3: Kommunikationsverlauf im OAuth-Protokoll 13
Abbildung 4: Gegenüberstellung zwischen einem normalen SSL-Handshake und einem umgekehrten SSL-Handshake 17

Tabellenverzeichnis

Tabelle 1: Statistik zu Authentifizierungsverfahren zum Schutz von Daten und Geräten 14
Tabelle 2: Statistik zur Frequenz, in der Nutzer ihre Passwörter wechseln 15
Tabelle 3: Parameter linearer Anpassungen an HTTP- und HTTPS-Übertragungen . 16
Tabelle 4: Demographie der Studie zur Entwicklung einer sicheren HTTPS-Verbindung 19

Abkürzungsverzeichnis

2FA Zwei-Faktor Authentifizierung
AES Advanced Encryption Standard
API Application Programming Interface
BSI Bundesamt für Sicherheit in der Informationstechnik
DEA Data Encryption Algorithm
DES Data Encryption Standard
HMAC Hashed Message Authentication Code
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
IBM International Business Machines Corporation
JS JavaScript
JWT JSON Web Token
MC Master Key
MITM Man-In-The-Middle
NIST National Institute of Standards and Technology
NSA National Security Agency
MD5 Message-Digest Algorithm 5
OAuth Open Authorization
OTP One-Time Password
RFC Request for Comments
RSA Rivest-Shamir-Adleman
SHA Secure Hash Algorithm
SSL Secure Sockets Layer
SSO Single-Sign-On
TCP Transmission Control Protocol
TLS Transport Layer Security

1 Einleitung

Die Sicherheit von Daten in der digitalen Welt ist von größter Bedeutung. Das Internet und Webanwendungen haben unser Leben verbessert, aber auch neue Herausforderungen in Bezug auf Datensicherheit gebracht.

Kryptografie ermöglicht durch Verschlüsselungs- und Signaturverfahren eine sichere Übertragung und Speicherung von Daten. Sie sichert Daten und Informationen, die über das Internet oder andere Netzwerke übertragen werden, mit mathematischen Verfahren. Verschlüsselung gewährleistet, dass nur autorisierte Personen die Informationen lesen können, während digitale Signaturen Manipulation erkennbar machen.

1.1 Zielsetzung

In dieser Arbeit wird untersucht, wie Verschlüsselung und digitale Signaturen als mathematische Methoden eingesetzt werden, um Webanwendungen zu sichern. Webanwendungen werden immer häufiger zur Speicherung und Verarbeitung von verschiedensten Daten und Informationen verwendet und stellen daher ein attraktives Ziel für Angreifer dar. Durch den Einsatz von Kryptografie kann die Sicherheit von Webanwendungen erhöht werden.

Dabei wird versucht, die Forschungsfrage „Wie sehr tragen die aktuell genutzten kryptografischen Methoden zur Sicherheit von Daten in Webanwendungen bei?“ anhand einer Literaturanalyse zu beantworten.

1.2 Aufbau der Arbeit

Abschnitt 2 erklärt einige grundlegende Konzepte der Kryptografie, und stellt verschiedene kryptografische Verfahren und Algorithmen dar. Abschnitt 3 befasst sich mit verschiedenen Möglichkeiten, Verschlüsselungsverfahren in Webanwendungen einzubinden und die Sicherheit von Daten zu gewährleisten, während Abschnitt 4 Probleme und Schwierigkeiten, die während und nach der Implementierung dieser Verfahren auftreten können, erläutert.

2 Grundlagen der Kryptografie - Kryptografische Verfahren und Algorithmen

In der digitalen Welt spielt die Kryptografie eine entscheidende Rolle. Sie sichert Informationen durch Ver- und Entschlüsselung, um Vertraulichkeit, Integrität und Authentizität zu gewährleisten. kryptografische Algorithmen dienen als grundlegende Bausteine, um Daten in eine unlesbare Form zu bringen und nur autorisierten Empfängern zugänglich zu machen.

Dieser Abschnitt gibt einen Überblick über die Grundlagen der Kryptografie und verschiedene Arten von Algorithmen, die in Webanwendungen und in der Informationssicherheit verwendet werden. Es vermittelt die Prinzipien und Konzepte, um das Verständnis für die Bedeutung und den Einsatz von Kryptografie in der vernetzten Welt zu stärken.

2.1 Symmetrische Verschlüsselungsalgorithmen - Data Encryption Standard (DES)

Bei symmetrischen Verschlüsselungsalgorithmen wird, wie in Abbildung 1[1] dargestellt, derselbe Schlüssel zum Ver-und Entschlüsseln einer Nachricht verwendet. Dies ermöglicht eine einfache und schnelle Kommunikation, führt jedoch dazu, dass die Geheimhaltung des Schlüssels schnell zu einer Sicherheitslücke führen kann.

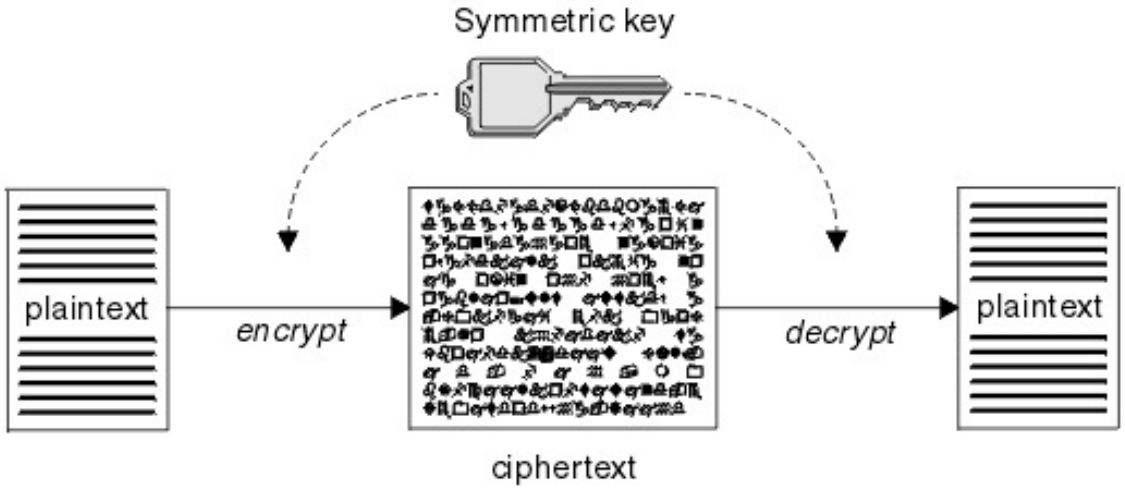


Abbildung 1: Schaubild einer symmetrischen Verschlüsselung[2]

[1] *Krawisz, D.*, 2013.
[2] ebd.

Symmetrische Verschlüsselungsverfahren werden deshalb dann als sicher angesehen, wenn man ohne den Schlüssel den Klartext nicht aus dem verschlüsselten Text ermitteln kann.[3]

DES, oft auch Data Encryption Algorithm (DEA), wurde 1975 im *Federal Register* der USA veröffentlicht und als Kooperation zwischen dem National Institute of Standards and Technology (NIST), der National Security Agency (NSA) und International Business Machines Corporation (IBM) entwickelt.[4]

DES ist ein Blockchiffre-Verfahren, was bedeutet, dass die Daten in immer gleich großen Blöcken, hier in Blöcken von 64 Zeichen, und mit einer immer gleichen Schlüssellänge, hier 56 Bit, verschlüsselt werden.[5]

2.2 Asymmetrische Algorithmen

Asymmetrische Verschlüsselungsverfahren verwenden, im Gegensatz zu den Symmetrischen, welche im vorherigen Abschnitt behandelt sind, zwei Schlüssel. Einen Public Key, zum Verschlüsseln der Daten und einen Private Key, zum Entschlüsseln. Dabei muss ausschließlich der Private Key geheim gehalten werden.

2.2.1 RSA-Verfahren

Rivest-Shamir-Adleman (RSA) ist das erste entwickelte Public-Key Verschlüsselungsverfahren und besitzt auch heute noch eine große Relevanz.[6]

2.2.1.1 Schlüsselerzeugung

Das Generieren des RSA-Schlüsselpaars benötigt einige verschiedene Zahlen. Der Sicherheitsparameter $k \in \mathbb{N}$ gibt die Größe des Produkts der beiden gewählten Primzahlen für die Verschlüsselung an. Zwei statistisch unabhängige, zufällige Primzahlen p und q werden ausgewählt, um den RSA-Modul n zu bilden. Dieser Wert wird später für Ver- und Entschlüsselung verwendet und berechnet sich als $n = p * q$.

Zusätzlich wird eine natürliche, ungerade Zahl e gewählt, für die

[3] *Thielert*, *S.*, 2007, S. 5.
[4] *Johnson*, *T. R.*, 1998, S. 232.
[5] *Thielert*, *S.*, 2007, S. 6.
[6] *Buchmann*, *J.*, 2016, S. 168.

$$1 < e < \varphi(n) = (p-1)(q-1) \text{ und } \gcd(e,(p-1)(q-1)) = 1 \tag{1}$$

gilt und daraus mit den folgenden Bedingungen

$$1 < d < (p-1)(q-1) \text{ und } d * e \equiv 1 \mod (p-1)(q-1) \tag{2}$$

eine weitere Zahl $d \in \mathbb{N}$ gebildet.

Da $\gcd(e,(p-1)(q-1)) = 1$ gilt, existiert eine solche Zahl d definitiv. Berechnet werden kann sie mit dem erweiterten euklidischen Algorithmus. Der Public Key bildet sich dabei aus dem Paar (e,n), der Private Key aus der Zahl d.[7]

Damit RSA eine sichere Verschlüsselung ermöglichen kann, müssen die beiden Primzahlen p und q passend gewählt werden. Dafür ist es üblich, dass k als eine gerade, mindestens 2^{10}-Bit lange Zahl gewählt wird.[8]

2.2.1.2 Verschlüsselung

Um mit dem RSA Algorithmus eine Nachricht zu verschlüsseln, wird der Public Key (e,n) benötigt. Aus einem Klartext $m \in \mathbb{Z}_m$ mit $\mathbb{Z}_m$ als RSA-Klartextraum erhält man den verschlüsselten Text c mit

$$c = m^e \mod n \tag{3}$$

m kann wieder rekonstruiert werden mit

$$m = c^d \mod n \tag{4}$$

wobei c der zuvor erhaltene verschlüsselte Text, d der Private Key und n ein Teil des public Keys ist.[9]

2.2.2 Hashfunktionen — Secure Hash Algorithm (SHA)

Im Allgemeinen sind Hashfunktionen Algorithmen, die einen Text beliebiger Länge zu einem neuen Text mit vorgegebener Länge komprimieren.[10] Sie werden mithilfe von sog. Kompressionsfunktionen generiert.

---

[7] *Buchmann*, *J.*, 2016, S. 169.
[8] Ebd., S. 169.
[9] *Davis*, *T.*, 2023, S. 6.
[10] *Preneel*, *B.*, 2003, S. 15.

Damit Hash- und Kompressionsfunktionen in der Kryptografie zur Authentifizierung, wie z. B. zur Speicherung von Passwörtern genutzt werden können, müssen sie noch verschiedene Kriterien erfüllen. Diese werden folgend erklärt:

Definition 2.1. *Eine Einweghashfunktion ist eine Funktion h, die folgende Bedingungen erfüllt:*[11]

1. *Die Beschreibung von h muss öffentlich bekannt sein und sollte keine geheimen Informationen erfordern (Erweiterung des Kerckhoff'schen Prinzips*[12]*).*
2. *Das Argument X kann von beliebiger Länge sein und das Ergebnis $h(X)$ hat eine feste Länge von n-Bits (mit $n \geq 64$).*
3. *Für gegebene h und X, muss die Berechnung von $h(X)$ einfach*[13] *sein.*
4. *Die Hashfunktion muss in dem Sinne monodirektional sein, dass es bei einem Y im Abbild von h schwer*[13] *ist, aus einer Nachricht einen bestimmten Hashwert zu generieren, und dass es schwer*[13] *ist, zwei Nachrichten zu finden, die den gleichen Hashwert teilen.*

Definition 2.1 verwendet die Begriffe „einfach“ und „schwer“, da heutzutage noch keine Algorithmen bekannt sind, die eine Einweghashfunktion schnell genug umkehren können.[14]

Secure Hash Algorithms (SHAs) sind verschiedene kryptologische Hashfunktionen und eine Modifikation des Message-Digest Algorithm 5 (MD5), welche zur Berechnung eines Prüfwertes für beliebige Nachrichten dienen und unter anderem die Grundlage zur Erstellung einer digitalen Signatur, genauer erläutert in Unterunterabschnitt 3.2.1, sind.[15]

2012 wurde SHA-3[16] vom NIST standardisiert und wird heute als sicher angesehen,[17] aber auch die Algorithmen unter SHA-2 sind heute stark verbreitet. SHA-2 und SHA-3 bezeichnen nicht einzelne Algorithmen sondern Algorithmusgruppen, deren zugrunde liegende Algorithmen sich primär in der Länge des ausgegebenen Prüfwertes unterscheiden. Dabei werden die Algorithmen SHA-256 und SHA-512 am häufigsten genutzt.

---

[11] Vgl. *Preneel*, *B.*, 2003, S. 17.
[12] *Petitcolas*, *F.*, o. J.
[13] „einfach“ und „schwer“ sind hier im kryptografischen Sinne zu verstehen und beziehen sich auf das Zusammenspiel von Laufzeit und Rechenaufwand eines Algorithmus'
[14] *Buchmann*, *J.*, 2016, S. 234.
[15] *Encryption Consulting LLC*, 2023.
[16] Auch als *Keccak* bezeichnet
[17] *Buchmann*, *J.*, 2016, S. 239.

3 Anwendung von Kryptografie in Webanwendungen

Webanwendungen spielen in der heutigen vernetzten Welt eine bedeutende Rolle bei der Bereitstellung von Diensten und dem Austausch sensibler Informationen.

Dieser Abschnitt behandelt den Einsatz von Kryptografie in Webanwendungen, um die Sicherheitsanforderungen hinsichtlich Vertraulichkeit, Integrität und Verfügbarkeit zu erfüllen. Es werden die grundlegenden Sicherheitsanforderungen an Webanwendungen erläutert und gezeigt, wie Kryptografie genutzt werden kann, um sensible Daten vor unberechtigtem Zugriff zu schützen.

Ein zentraler Aspekt der Sicherheit von Webanwendungen ist die Verschlüsselung der Datenübertragung. Daher wird sich der nächste Abschnitt auf das Hypertext Transfer Protocol Secure (HTTPS)-Protokoll und die Secure Sockets Layer (SSL)/Transport Layer Security (TLS)-Verschlüsselung konzentrieren.

Die Sicherheit von Passwörtern wird ebenfalls behandelt, indem Methoden wie Salted Hashing und Key Derivation Functions vorgestellt werden, um Passwörter sicher zu speichern. Darüber hinaus werden verschiedene Verfahren und Protokolle zur Authentifizierung und Autorisierung in Webanwendungen erläutert, wie z. B. tokenbasierte Verfahren und OAuth, um die Identitätsprüfung und Zugriffskontrolle zu gewährleisten.

3.1 Sicherheitsanforderungen an Webanwendungen — Vertraulichkeit, Integrität, Verfügbarkeit

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) listet für die Webanwendung drei primäre Schutzziele

- Vertraulichkeit,
- Integrität sowie
- Verfügbarkeit

von denen zudem abgeleitet u. a. die beiden sekundäre Schutzziele

- Authentizität und
- Nicht-Abstreitbarkeit

aufgeführt sind.[18]

[18] *Bundesamt für Sicherheit in der Informationstechnik*, 2022, S. 8.

Das Zusammenarbeiten der Schutzziele, insbesondere der drei primären Schutzziele, zur Garantie einer sicheren Webanwendung wird im Folgenden untersucht.

3.2 Verschlüsselung von Datenübertragungen — SSL/TLS-Verschlüsselung, HTTPS-Protokoll

3.2.1 Digitale Signatur — SSL/TLS-Verschlüsselung

3.2.1.1 Digitale Signatur

Eine digitale Signatur ist ein Public-Key Verschlüsselungsverfahren, bei dem einer Nachricht ein einzigartiger Schlüssel angehangen wird. Dieser wird daraus generiert, dass der Hashwert einer Nachricht mit dem Private Key des Absenders verschlüsselt wird.[19] Der Klartext, die Signatur und der Public Key des Senders werden verpackt und mit dem Public Key des Empfängers zusammen verschlüsselt. Diese signierte und verschlüsselte Nachricht wird anschließend übermittelt.[20]

Um die Nachricht zu entschlüsseln wird, wie bei anderen Public-Key Verschlüsselungsverfahren, der Private Key des Empfängers entsprechend des genutzten Verfahrens auf die Nachricht angewandt. Anschließend wird der Klartext der Nachricht gehashed und die Signatur mit dem Public Key des Senders entschlüsselt. Diese beiden Werte werden auf Gleichartigkeit verglichen, in welchem Fall die Nachricht verifiziert ist.[21]

3.2.1.2 SSL/TLS-Verschlüsselung

Ein wichtiges Werkzeug der sicheren Webentwicklung sind SSL-und TLS-Zertifikate (Unterunterabschnitt 3.2.1). TLS, als Nachfolger für SSL, nutzt das gleiche X.509-Zertifikat.[22] Über TLS kann ein Client überprüfen, ob das Zertifikat des angefragten Servers von einer vertrauten Autorität ausgestellt wurde oder nicht, indem die X.509-Zertifikate verifiziert werden[23] und bietet somit eine „externe Überprüfung durch einen vertrauenswürdigen Dritten“.[24]

---

[19] *Kaur*, *R.*, *Kaur*, *A.*, 2012, S. 297.
[20] Ebd., S. 297.
[21] Ebd., S. 297.
[22] Aus Gründen der Übersichtlichkeit und der Ähnlichkeit der beiden Zertifikate, wird im Folgenden SSL/TLS vereinfachend als TLS bezeichnet
[23] Vgl. *Zhang*, *L.* et al., 2014, S. 1.
[24] *Cloudfare*, o. J., Grund № 3.

Das TLS-Protokoll besteht selbst aus weiteren Unterprotokollen, die sich gegenseitig unterstützen und aufeinander aufbauen.

Eines der TLS-Unterprotokolle ist das TLS-Handshake-Protokoll, dessen Verlauf in Abbildung 2[25] dargestellt wird.

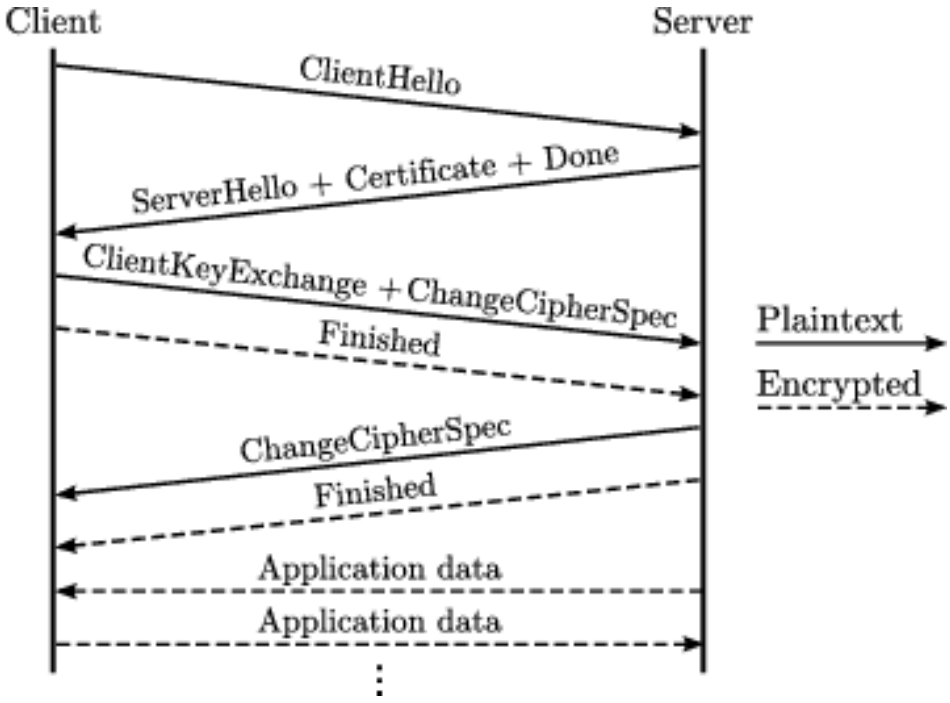


Abbildung 2: Darstellung eines TLS-Handshakes[26]

Nachdem der Client bei Verbindungsaufbau dem Server eine Nachricht schickt, antwortet Server mit einer Nachricht und seinem Zertifikat. Dies wird vom Client verifiziert, woraufhin er seine Schlüssel und die Daten seines Ciphers, also unter anderem den genutzten Verschlüsselungsalgorithmus übermittelt. Der Server reagiert damit, seine Cipherdaten zu übermitteln, wodurch der Handshake abgeschlossen ist und die Anwendungsdaten gesendet werden.

Mithilfe dieser Handshakes wird ausgehandelt, welche kryptografischen Algorithmen und Schlüssel verwendet werden und die Kommunikationspartner identifizieren und authentifizieren sich, hierbei ist es üblich, dass sich nur der Server bei dem Client authentifiziert, was Ein-Weg Authentifizierung bezeichnet wird.[27]

[25] Aus *Husák, M.* et al., 2015, S. 2.
[26] Aus ebd., S. 3
[27] Vgl. *Morrissey, P., Smart, N. P., Warinschi, B.*, 2010, S. 191.

3.2.2 HTTPS-Protokoll

Wie in Unterunterabschnitt 3.2.1 bereits dargestellt, sorgt TLS dafür, dass eine Anwendung oder ein Nutzer verifizieren kann, dass die angefragten Daten von dem Absender stammen, den man angefragt hat. Dies trägt besondere Relevanz bei Webanwendungen, da es sonst möglich ist, unter falschem Namen schädliche Software in ein System einzuspeisen. Das Hypertext Transfer Protocol (HTTP) überträgt Daten über das Transmission Control Protocol (TCP), wohingegen HTTPS die Daten verschlüsselt über das TLS-Protokoll verschickt.[28]

Eine HTTPS-Verschlüsselung einer Internetseite sorgt dafür, dass bei der Übertragung Daten, besonders vertrauliche wie z. B. Bankkontoinformationen, nicht von dritten eingesehen werden können.[29] Zudem lässt das TLS-Zertifikat den Nutzer einsehen, dass der Absender der Daten das angefragte System ist, wie in Unterunterabschnitt 3.2.1 dargestellt. Dadurch wird verhindert, dass ein Angreifer einem Nutzer eine identisch aussehende Internetseite bereitstellt und dadurch die Nutzer glauben lässt, sie seien auf der gewünschten Internetseite.[30]

3.3 Passwortsicherheit — Schlüsselableitung, Salted- & Peppered-Hashing

3.3.1 Schlüsselableitung

Als Schlüsselableitung wird in der Kryptografie eine Operation bezeichnet, welche aus einem Schlüssel oder einem Passwort verschiedene andere Schlüsselwerte generiert.[31] Für diese Operation gibt es verschieden Möglichkeiten, zwei häufig angewandte sind Salted- (Unterunterabschnitt 3.3.2) und Peppered-Hashing (Unterunterabschnitt 3.3.3), bzw. das kombinieren dieser Verfahren. Der so generierte Schlüssel wird Master Key (MC) gennant.

3.3.2 Salted Hashing

Als Salt wird in der Kryptografie ein zufälliger String bezeichnet, der einem zu verschlüsselnden Klartext vor dem Hashprozess angehangen wird.[32] Dabei wird für jede Zeile der

[28] *Naylor*, *D.* et al., 2014; *Dierks*, *T.*, *R.*, *R.*, 2008.
[29] Vgl. *Cloudfare*, o. J.
[30] Vgl. ebd.
[31] Vgl. *Sönmez Turan*, *M.* et al., 2010, S. 3.
[32] Vgl. *Rosulek*, *M.*, 2021, S. 205.

Datenbank ein eigener Salt generiert und in der Datenbank gespeichert. Dies verhindert u. a., dass für zwei Datensätze mit identischem Klartext der gleiche Hashwert in der Datenbank gespeichert wird.

Dadurch werden Rainbow-Tables für die Datenbank überflüssig, da jeder einzelne Eintrag eine Rainbow-Table benötigt und nicht nur die Datenbank als ganzes.[33]

Für eine ausreichende Sicherheit muss ein Salt zwei Anforderungen erfüllen. Einzigartigkeit wurde weiter oben bereits aufgeführt. Zudem wird der Datensatz sicherer, umso länger der Salt ist. Der NIST-Standard für Schlüsselableitung sieht dabei vor, dass der MC, mindestens eine Länge von 112 Bits haben soll.[34]

3.3.3 Peppered Hashing

Analog zum Salted Hashing (Unterunterabschnitt 3.3.2) gibt es die Sicherungsmethode des Pepper, bzw. des Peppered Hashing.

Bei diesem wird, wie beim Salted Hashing auch, ein zufällig generierter String an den Klartext angehangen, bevor der Hashalgorithmus durchgeführt wird. Dieser String wird einmalig bei der Einrichtung des Servers festgelegt und anschließend geheimgehalten.[35] Auch dieses Verfahren schützt vor einer möglichen Rainbow-Table. Gleichermaßen gibt es beim Peppered Hashing keine festgelegte Länge, welche der Pepper aufweisen sollte, es empfiehlt sich jedoch auch hier, einen möglichst langen Wert zu nehmen.

3.4 Authentifizierung und Autorisierung — Token-Verfahren, OAuth

Um Datensicherheit zu gewährleisten, bzw. Datenzugriff zu regulieren, werden in Webanwendungen zwei Konzepte verfolgt: Authentifizierung und Autorisierung.

Autorisierung befasst sich mit der Zugriffskontrolle darüber, welche Geräte oder Nutzer welche Daten lesen und/oder schreiben, Programme ausführen oder Akteure kontrollieren können,[36] oder diesen Zugriff z. B. wegen Malware zu entfernen.[37]

Demgegenüber bezeichnet Authentifizierung den Prozess, einen Client, Gerät oder Mensch, zu identifizieren und ist eine Grundvoraussetzung für Autorisierungen, da in den

[33] Vgl. *Rosulek*, *M.*, 2021, S. 205.
[34] Vgl. *Sönmez Turan*, *M.* et al., 2010, S. 6.
[35] Vgl. *Webster*, *C. R.*, 2009.
[36] Vgl. *Kim*, *H.*, *Lee*, *E. A.*, 2017, S. 28.
[37] Vgl. ebd., S. 28.

meisten Fällen Autorisierung ohne eine zuvor gehende Authentifizierung nicht möglich ist.[38] Häufig genutzte Methoden zur Authentifizierung sind die zuvor behandelten digitalen Zertifikate (Unterunterabschnitt 3.2.1), das HTTPS-Protocol (Unterunterabschnitt 3.2.2), sowie Schlüsselableitung (Unterunterabschnitt 3.3.1). Diese werden genutzt, um das Vertrauen zwischen dem Client und der Anwendung herzustellen.[39]

3.4.1 Access Token

Eine häufig genutzte Art, die Autorisierung eines Clients zu prüfen ist es, einen Access Token zu generieren. Diese bestehen üblicherweise aus drei Schlüsselelementen,

- dem Header, mit Daten über die Art des Tokens und den Algorithmus, mit dem dieser generiert wurde,
- der Payload, welche die Informationen über den Client beinhält, darunter u. a. die Berechtigungen und Ablaufdaten, sowie
- die Signatur, ein Zertifikat (Unterunterabschnitt 3.2.1), das die Echtheit des Tokens garantiert;

zusätzlich können weitere Elemente wie z. B. Metadaten angefügt sein, um fallspezifische Anforderungen zu erfüllen.[40]

Eine häufige Variante von Access Token ist der JSON Web Token (JWT), der aufgrund seiner geringen Speichergröße besonders beliebt ist.[41] Der JWT besteht dabei aus den oben beschriebenen Elementen, die, bevor sie mit einem '.' zusammengefügt werden, mit einem Base64-Algorithmus kodiert und komprimiert werden.[42]

Ein Minimalcodebeispiel um in der Programmiersprache JavaScript (JS) einen JWT zu generieren, sieht wie folgt aus

Listing 1: Beispielskript zur Generierung eines JWT

```
1 import {sha256} from "js-sha256";
2 import base64url from "base64url";
3
4 const secret = "private-key"
5
6 const header = {
7         "alg": "HS256",
```

[38] Vgl. *Kim*, *H.*, *Lee*, *E. A.*, 2017, S. 28.
[39] Vgl. ebd., S. 28.
[40] Vgl. *Okta*, 2023.
[41] Vgl. *Jones*, *M. B.*, *Bradley*, *J.*, *Sakimura*, *N.*, 2015, S. 4.
[42] Vgl. ebd., S. 5.

```
        "typ": "JWT"
}
const payload = {
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
const signature = sha256.hmac(
        secret,
        base64url(header) + '.' +
        base64url(payload)
)

const token = base64url(header) + '.' + base64url(payload) + '.' + base64url(signature
    )
```

Hierbei stellt `secret` den Private Key[43] des digitalen Zertifikates (Unterunterabschnitt 3.2.1), `base64url()` eine Funktion, Daten mit einem Base64-Algorithmus zu kodieren und `sha256.hmac()` eine Methode, um einen Hashed Message Authentication Code (HMAC) mit dem SHA256-Algorithmus zu erzeugen, dar.

Dabei wird der Token

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.
    SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c"
```

generiert.

3.4.2 Open Authorization (OAuth)

OAuth bezeichnet ein Framework, welches einem Dateninhaber ermöglicht, Dritten begrenzten Zugriff zu gewähren, z. B. bei Single-Sign-Ons (SSOs), oder den gegebenen Zugriff zu beschränken.[44]

Um Zugriff anzufragen, bzw. ihn zu überprüfen[45] verwendet das OAuth Protokoll die folgenden 6 Elemente:

- einen Client als den Service, der nach Autorisierung fragt,
- einem Ressourceninhaber als die Entität, die die Information, für die der Client Zugriff anfragt,

[43] *Krawczyk*, *D. H.*, *Bellare*, *M.*, *Canetti*, *R.*, 1997.
[44] Vgl. *Hardt*, *D.*, 2012, S. 1; Vgl. *Leiba*, *B.*, 2012, S. 75.
[45] Vgl. *Leiba*, *B.*, 2012, S. 75.

- einen Ressourcenserver als den Service, der den Zugriff zu den angefragten Informationen vergibt,
- einen Autorisierungsserver, der die Echtheit der Zertifikate des Ressourceninhabers verifiziert und die Überprüfungen für die Autorisierung durchführt,
- einen Access Token (Unterunterabschnitt 3.4.1), der von dem Autorisierungsserver generiert wird und dem Client den Zugriff von dem Ressourcenserver anfragen lässt und
- einen Authentifizierungscode, den der Autorisierungsserver während der Anfrage überprüfen kann

Abbildung 3[46] stellt die Kommunikation zwischen dem Client und der Service-Application Programming Interface (API) dar. Zunächst stellt der Client eine Anfrage

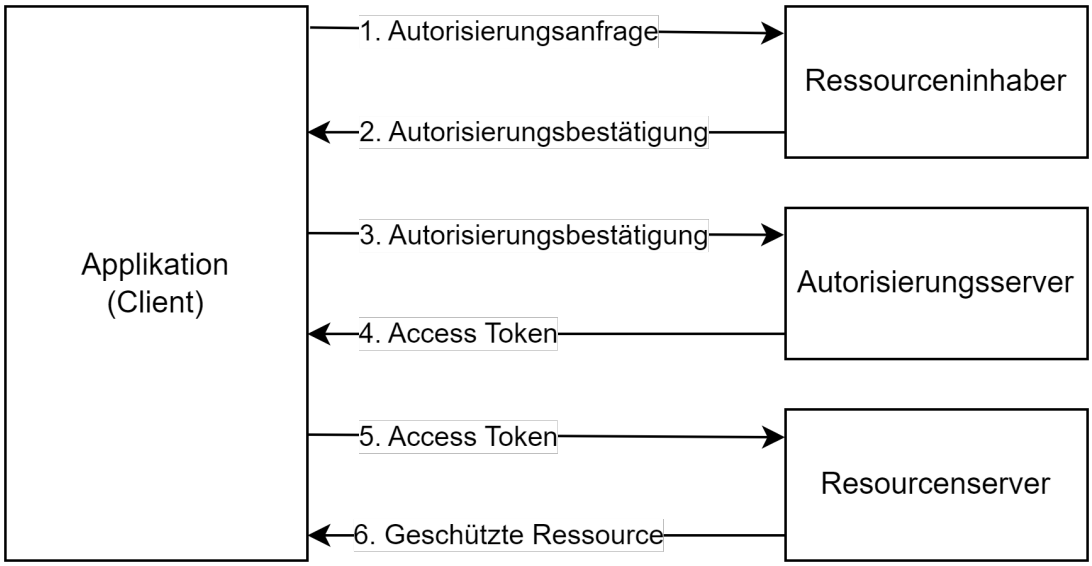


Abbildung 3: Kommunikationsverlauf im OAuth-Protokoll[47]

an den Ressourceninhaber nach einer Autorisierung. Wenn diese dem Client bestätigt wird, stellt dieser mit der Bestätigung eine Anfrage nach einem Access Token an den Autorisierungsserver. Mit dem Access Token stellt der Client zuletzt eine Anfrage an den Ressourcenserver und erhält von diesem die gewünschten Ressourcen.

OAuth wird primär für SSOs genutzt, um z. B. einer Anwendung Zugriff auf bestimmte Daten einer anderen Anwendung zu gewähren, ohne die Anmeldedaten direkt zu übergeben, wie etwa Bilder aus einem Cloudservice an einen Druckservice zu übergeben.[48]

[46] Vgl. *Hardt*, *D.*, 2012, S. 7, Abbildung 1.
[47] Vgl. ebd., S. 7, Abbildung 1
[48] Vgl. *Leiba*, *B.*, 2012, S. 74.

3.4.3 Zwei-Faktor Authentifizierung (2FA)

Wie Tabelle 1[49] zeigt, sind Passwörter heutzutage mit Abstand die am weitesten verbreitete Methode, sich bei einem Service zu authentifizieren.

Tabelle 1: Statistik zu Authentifizierungsverfahren zum Schutz von Daten und Geräten[50]
Details: Deutschland; 08. bis 14.08.2018; 1.025 Befragte; ab 18 Jahre

Merkmal	Anteil der Befragten
Ich nutze ein Passwort	63 %
Ich nutze einen Fingerabdruckssensor	26 %
Ich nutze eine Zahlenkombination	22 %
Ich nutze ein Muster	12 %
Ich nutze eine Gesichtserkennung	9 %
Ich nutze eine Spracherkennung	8 %
ich nutze die Online-Ausweisfunktion meines Personalausweises	7 %
Ich nutze einen Iris-Scanner	7 %
Ich nutze einen Venen-Scanner	5 %
Sonstiges	0 %
Ich nutze keine Authentifizierungsverfahren	12 %

Jedoch bietet ein einzelnes Passwort als Authentifizierungsverfahren keinen vollständigen Schutz,[51] da Angreifer z. B. durch andere unsichere Software auf dem System einfach an die Nutzerdaten gelangen können. Besonders, wenn man für mehrere verschiedene Systeme das gleiche Passwort nutzt, einen SSO-Service für mehrere Dienste nutzt oder, wie in Tabelle 2[52] dargestellt, die Passwörter nur selten wechselt, sind diese Angriffen deutlich stärker ausgesetzt.

2FA-Dienste tragen dazu bei, dass diese Unsicherheiten abgeschafft werden, in dem z. B. ein zusätzliches, verknüpftes Gerät einen Authentifizierungsvorgang startet, welcher bestätigt werden muss, wie es unter anderem verschiedene Banksysteme machen, oder ein externer Dienst für 2FA-One-Time Passwörter (OTPs) genutzt wird, die in bestimmten Zeitintervallen Codes generieren, der zusätzlich zu der primären Authentifizierungsmethode eingegeben werden muss.

---

[49] *Statista*, 2019.
[50] ebd.
[51] Vgl. *Jacomme*, *C.*, *Kremer*, *S.*, 2021, S. 2.
[52] *Bitwarden*, 2022.
[53] ebd.

Tabelle 2: Statistik zur Frequenz, in der Nutzer ihre Passwörter wechseln[53]
Details: Weltweit; 2022; >2000 Befragte

Charakteristik	Anteil
Jeden Tag	6%
Mehrfach die Woche	15%
Ungefähr einmal im Monat	34%
Selten	44%

4 Herausforderungen bei der Implementierung von kryptografischen Funktionen in Webanwendungen

In vielen Fällen überwiegt der Schutz, den kryptografische Methoden bieten, den Aufwand und die Probleme, die bei der Implementierung auftreten. Dennoch wird versucht, die Probleme, wie z. B. die in Unterabschnitt 4.1 beschriebenen Leistungseinbußen oder die in Unterabschnitt 4.3 erläuterte erschwerte Benutzbarkeit, zu beheben, bzw. auf ein Minimum zu reduzieren.

4.1 Performance- und Skalierbarkeitsprobleme

Der in Unterunterabschnitt 3.2.2 vorgestellte Vorteil der verschlüsselten Übertragung bringt auch einen Nachteil mit sich. Die Verschlüsselung der Daten verringert die Übertragungsrate und verlängert die Latenz der Übertragung,[54] dies wird in Tabelle 3[55] dargestellt.

Diese Daten wurden vor 25 Jahren erfasst und seitdem ist HTTPS verbessert, wodurch der nötige Performanceaufwand um eine verschlüsselte Verbindung aufzubauen, verringert wurde.[56]

Neuere Versionen des TLS-Algorithmus bringen noch deutlichere Performance-Verbesserungen mit sich, da unter anderem nur ein TLS-Handshake, vorgestellt auf Seite 8, benötigt wird, bzw. keiner, wenn bereits eine Verbindung zwischen Client und Server bestand.[57]

[54] *Goldberg*, *A.*, *Buff*, *R.*, *Schmitt*, *A.*, 1998, S. 5.
[55] Übersetzt nach: Ebd., S. 5.
[56] Vgl. *Cloudfare*, o. J.
[57] Vgl. ebd.

Tabelle 3: Parameter linearer Anpassungen an HTTP- und HTTPS-Übertragungen[58]

	Transferrate (bytes/ms)		Latenz (ms)	
Server	Netscape	Mircosoft	Netscape	Microsoft
Unsicher	946	829	4.5	19.0
Sicher 40 bit	730	689	3.6	3.9
Sicher 128 bit	736	686	25.0	5.1

Diese Daten wurden vor 25 Jahren erfasst. Seitdem wurde HTTPS verbessert, wodurch sich der nötige Leistungsaufwand, um eine verschlüsselte Verbindung aufzubauen, verringerte.[59]

Gleichermaßen ist, trotz des Verdoppelns von CPU Geschwindigkeiten alle 18 Monate, die Performance von SSL Maschinen noch immer ein Problem.[60]

Gemäß Abbildung 4[61] kann die Authentifizierung der Client-Identität durch Entschlüsselung und Verifizierung des Zertifikats und der Signatur erfolgen. Eine Möglichkeit, dies zu erreichen, besteht darin, die Kommunikationsrichtung der Handshakes umzukehren. Dies bedeutet, dass die Daten nun serverside anstatt clientside verifiziert und entschlüsselt werden.[62]

Da ein Großteil der Berechnungen für einen TLS-Handshake während des Verifizierens der Zertifikate durchgeführt werden, wird die Laufzeit des Handshakes verbessert, wenn diese Berechnungen vom Server durchgeführt werden.

[58] *Goldberg*, *A.*, *Buff*, *R.*, *Schmitt*, *A.*, 1998, S. 5
[59] Vgl. *Cloudfare*, o. J.
[60] Vgl. *Bicakci*, *K.*, *Crispo*, *B.*, *Tanenbaum*, *A. S.*, 2003, S. 2.
[61] Ebd., S. 3.
[62] Vgl. ebd., S. 2.
[63] ebd., S. 3

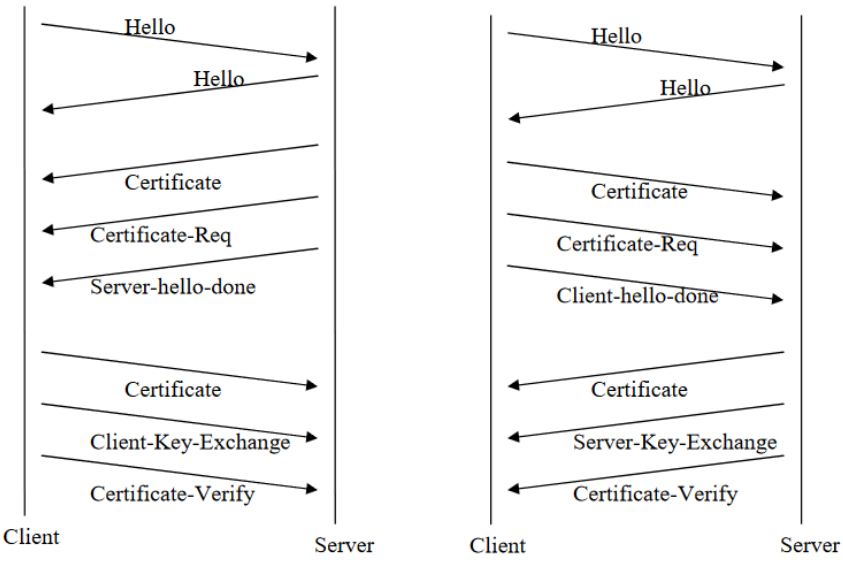


Abbildung 4: Gegenüberstellung zwischen einem normalen SSL-Handshake (links) und einem umgekehrten SSL-Handshake (rechts)[63]

4.2 Komplexität und Fehleranfälligkeit der kryptografischen Methoden

4.2.1 Komplexität bei der Implementierung von kryptografischen Methoden in Webanwendungen

Durch die Vielzahl an Lektüre zu den verschiedenen Verschlüsselungsmethoden[64] ist es heutzutage nicht mehr so kompliziert, kryptografische Methoden in Webanwendungen einzubauen, um Daten und die Anwendung als Ganzes zu sichern. Für viele Programmiersprachen sind heutzutage vorgefertigte Methoden oder Packages entwickelt worden, welche verschiedene kryptografische Methoden bereitstellen oder bündeln. So stellt die Programmiersprache JS die Funktion `digest(algorithm, data)` zur Verfügung, welche einen Datensatz nach einem SHA-Algorithmus[65] verschlüsselt.[66] Die Funktion `encrypt(algorithm, key, data)` hingegen erlaubt es, mit RSA, beschrieben in RSA oder Advanced Encryption Standard (AES) Algorithmen zu verschlüsseln.

So stellen gerade Side-Channel Angriff eine Gefahr da. Diese brechen den kryptografischen Algorithmus, indem sie Informationen aus dem Laufzeitverhalten analysieren. Durch u. a. der Ausführungsdauer des Algorithmus oder der abgestrahlten elektromagnetischen Strahlung der Hardware können diese gewonnen werden.[67]

[64] z. B. *Davies*, *J.*, 2011.
[65] SHA-1, SHA256, SHA-384 oder SHA-512
[66] Vgl. *MDN Web Docs*, o. J.
[67] Vgl. *Köpf*, *B.*, *Basin*, *D.*, 2007, S. 286.

4.2.2 Fehleranfälligkeit von kryptografischen Methoden in Webanwendungen

Trotz der vielfältigen analysen des TLS-Protokolls,[68] sind heute nur geringe Mängel und Schwachstellen bekannt.[69]

Definition 4.1 (Man-In-The-Middle (MITM))**.** *MITM bezeichnet nach Request for Comments (RFC) №2828[70] eine Form des aktiven Lauschangriffs, bei dem der Angreifer kommunizierte Daten abfängt und selektiv verändert, um sich als eine oder mehrere der an einer Kommunikationsbeziehung beteiligten Einheiten auszugeben.*

In einem typischen MITM Angriff stellt sich das angreifende System so zwischen den Client und den Server, sodass es mit den Client und den Server unabhängig voneinander kommunizieren kann, die beiden Parteien aber den Eindruck behalten, sie würden direkt miteinander kommunizieren; eine Möglichkeit, sich einen MITM Angriff vorzustellen ist es, als würde zwischen Client und Server ein SSL/TLS Proxy Server geschaltet sein.[71] Dadurch wissen weder der Client noch der Server über den MITM bescheid. Viele kryptografische Methoden tragen bei einem MITM Angriff nicht mehr zu der Sicherheit bei, da der MITM in der Verbindung integriert ist und alle Daten abfangen kann, so werden z. B. Anmeldedaten dem MITM sichtbar gemacht.[72] Das durchführen von MITM Angriffen wird jedoch durch bestimmte Zertifikatinfrastrukturen und Zertifizierungsstellen erschwert, wenngleich die Kontrolle über eine Zertifizierungsstelle den Angreifern die Möglichkeit bietet, jede beliebige Domäne anzugreifen.[73]

4.3 Benutzerfreundlichkeit und Usability-Aspekte

Eine Studie aus 2017 unter 28 IT-Fachleute stellte sich der Frage, wie benutzerfreundlich das Implementieren und Veröffentlichen einer sicheren HTTPS-Verbindung zu einem Apache Web Server ist, um eine Sicherheitsprüfung zu bestehen.[74] Die Demographie der Studie wird in Tabelle 4[75] detaillierter dargestellt.

---

[68] Siehe z. B. *Krawczyk*, *H.*, *Paterson*, *K. G.*, *Wee*, *H.*, 2013; *Paulson*, *L. C.*, 1999; *Dowling*, *B.* et al., 2015; *Cremers*, *C.* et al., 2017.
[69] Vgl. *Oppliger*, *R.*, *Hauser*, *R.*, *Basin*, *D.*, 2006, S. 2239.
[70] Übersetzt aus *Shirey*, *D. R.*, 2000, S. 105.
[71] Vgl. *Sounthiraraj*, *D.* et al., 2014, S. 4.
[72] Vgl. ebd., S. 4.
[73] Vgl. *Holz*, *R.* et al., 2012, S.
[74] Vgl. *Krombholz*, *K.* et al., 2017, S. 1341.
[75] Vgl. ebd., S. 1342.

Tabelle 4: Demographie der Studie zur Entwicklung einer sicheren HTTPS-Verbindung[76]

Demografie	Anzahl	Prozent
Geschlecht		
Weiblich	2	10%
Männlich	26	90%
Alter		
Minimal	21	
Maximal	32	
Median	23	
Monate in der IT-Branche gearbeitet		
Minimal	2	
Maximal	120	
Median	25	
Erfahren als Systemadministrator		
Ja	17	60%
Nein	11	40%
Zuvor TLS konfiguriert		
Ja	17	60%
Nein	11	40%
Aktuell administrierend		
Firmenwebserver	5	17%
Privater Webserver	17	83%

In der Studie wurden verschiedene Aspekte in die Bewertung mit einbezogen, darunter die Anzahl an Fehlermeldungen, die Schlüsselgröße, ob der Private Key verschlüsselt war und welche Version von SSL oder TLS verwendet wurde. Die Studie ergibt, dass das Einrichten einer sicheren HTTPS-Verbindung, inklusive TLS-Zertifikat, selbst bei Studierenden, die bereits erfolgreich IT-Sicherheitskurse bearbeitet haben und ihr technisches Wissen in einer Umfrage bestätigt haben, durch verschiedene Ursachen, wie z. B. unklare Fehlermeldungen, verwirrende Dateistrukturen bei den Konfigurationsdateien und einem zu hohem Ausmaß an erfordertem Hintergrundwissen, deutlich erschwert wird.[77] Gleichermaßen sind bereits bestehende TLS-Konfigurationen oftmals nicht sicher und schützen Nutzer daraus folgend nicht zuverlässig vor MITM-Angriffen.[78]

[76] Vgl. *Krombholz, K.* et al., 2017, S. 1342
[77] Vgl. ebd., S. 1347.
[78] Vgl. ebd., S. 1350.

5 Fazit

In der vorliegenden Arbeit wurde das Thema Kryptografie in Webanwendungen untersucht. Folgende Fragestellung lag der Untersuchung zugrunde: Wie sehr tragen die aktuell genutzten kryptografischen Methoden zur Sicherheit von Daten in Webanwendungen bei?

Mithilfe einer ausgiebigen Literaturanalyse konnten dafür verschiedene kryptografische Algorithmen untersucht und daraus gebildete Techniken analysiert werden.

Die Sicherheit von Daten in Webanwendungen wird drastisch durch ein Zusammenspiel verschiedener einfacher Algorithmen erhöht und noch weiter durch daraus gebildete Techniken, wie z. B. digitale Zertifikate, verstärkt. Diese Arbeit zeigt auch, dass ein Großteil der Nutzer von Webanwendungen nur ein Passwort nutzt, um ihre Daten zu sichern, sowie dass viele Nutzer ihre Passwörter nur unregelmäßig wechseln. Die Kombination dieser beiden Fakten stellt bei den meisten Systemen ein großes Sicherheitsrisiko dar, da viele Systeme z. B. nicht ausreichend auf die Möglichkeit hinweisen, 2FA-Services zu nutzen und einige Systeme Daten unverschlüsselt übertragen oder speichern. Dies ist jedoch ein rückgängiger Trend.

Die Analyse der verschiedenen kryptografischen Methoden im Zusammenhang mit den Problemen und Hindernissen, die diese auslösen, zeigt, dass es schwierig ist, eine Webanwendung ausgiebig zu sichern. Jedoch gibt es mehrere verschiedene Methoden, die Daten zu sichern, die auch miteinander agieren können und sich gegenseitig verstärken können. So ist es möglich, z. B. Passwörter zu hashen, bevor sie in einer Datenbank gespeichert werden und zudem noch die Verbindung mit einem TLS-Zertifikat abzusichern. Das Ziel dieser Arbeit, den Nutzen verschiedener kryptografischer Methoden vor- und ihre Auswirkungen darzustellen, wurde somit erreicht.

Die Untersuchung der einzelnen Möglichkeiten zeigt, dass beim Absichern von Daten oder Verbindungen auf viele einzelne Aspekte zu achten ist. Diese einzeln genauer zu untersuchen und Möglichkeiten zu erarbeiten, die Sicherheit von Webanwendungen zu vereinfachen könnte für eine weiterführende Auseinandersetzung interessant sein.

Zudem beschäftigt sich die vorliegende Arbeit ausschließlich mit den Auswirkungen der einzelnen Funktionen und nicht mit dem prinzipiellen Arbeitsablauf. Hier bleibt im Detail zu klären, wie die einzelnen Funktionen arbeiten und wie man sie entweder einzeln oder im Zusammenspiel mit anderen Funktionen optimieren kann.

Literaturverzeichnis

Bicakci, *Kemal*, *Crispo*, *Bruno*, *Tanenbaum*, *Andrew S.* (2003): Reverse SSL: Improved Server Performance and DoS Resistance for SSL Handshakes, in: o. O., 2003-06-26

Bitwarden (2022): Frequency of resetting passwords worldwide in 2022, Englisch, World Password Day 2022 - Global Survey, Bitwarden, o. O.: Statista, 2022-04-26, URL: https://www.statista.com/statistics/1303484/frequency-of-password-resets-worldwide/

Buchmann, *Johannes* (2016): Einführung in die Kryptographie, 6. Aufl., Springer-Lehrbuch, Berlin: Springer Spektrum, 2016, XXVI 330 Seiten

Bundesamt für Sicherheit in der Informationstechnik (2022): Leitfaden zur Entwicklung sicherer Webanwendungen, Empfehlungen und Anforderungen an die Auftragnehmer, in (2022)

Cremers, *Cas*, *Horvat*, *Marko*, *Hoyland*, *Jonathan*, *Scott*, *Sam*, *van der Merwe*, *Thyla* (2017): A comprehensive symbolic analysis of TLS 1.3, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, o. O., 2017, S. 1773–1788

Davies, *Joshua* (2011): Implementing SSL/TLS using cryptography and PKI, o. O.: John Wiley und Sons, 2011

Davis, *Tom* (2023): RSA Encryption, Englisch, o. O.: geometer, URL: http://www.geometer.org/mathcircles/RSA.pdf [Zugriff: 2023-05-19]

Dierks, *T.*, *R.*, *Rescorla* (2008), RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, o. O., 2008-08, URL: https://www.rfc-editor.org/rfc/rfc5246

Dowling, *Benjamin*, *Fischlin*, *Marc*, *Günther*, *Felix*, *Stebila*, *Douglas* (2015): A cryptographic analysis of the TLS 1.3 handshake protocol candidates, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, o. O., 2015, S. 1197–1210

Goldberg, *Arthur*, *Buff*, *Robert*, *Schmitt*, *Andrew* (1998): A COMPARISON OF HTTP AND HTTPS PERFORMANCE, in (1998)

Hardt, *D.* (2012), The OAuth 2.0 Authorization Framework, RFC 6749, USA, 2012-10

Holz, *Ralph*, *Riedmaier*, *Thomas*, *Kammenhuber*, *Nils*, *Carle*, *Georg* (2012): X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-Middle, in: *Foresti*, *Sara*, *Yung*, *Moti*, *Martinelli*, *Fabio* (Hrsg.), Computer Security – ESORICS 2012, o. O.: Springer Berlin Heidelberg, 2012, S. 217–234

Husák, *Martin*, *Cermak*, *Milan*, *Jirsik*, *Tomas*, *Celeda*, *Pavel* (2015): Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting, in: 10th International Conference on Availabiity, Reliability and Security, o. O., 2015-08, S. 8

Jacomme, *Charlie*, *Kremer*, *Steve* (2021): An Extensive Formal Analysis of Multi-Factor Authentication Protocols, in: ACM Trans. Priv. Secur. 24 (2021), Nr. 2

Johnson, *Tom R.* (1998): American Cryptology during the Cold War, 1945 - 1989, Book 3: Retrenchment and Reform, 1972-1980, Englisch, Bd. 5.3, 6, o. O., 1998, Kap. 19: The Rebirth of Intelligence during the Carter Administration, S. 232, 262 S.

Jones, *Michael B.*, *Bradley*, *John*, *Sakimura*, *Nat* (2015), JSON Web Token (JWT), RFC 7519, o. O., 2015-05, URL: https://www.rfc-editor.org/info/rfc7519

Kaur, *Ravneet*, *Kaur*, *Amandeep* (2012): Digital Signature, in: Conference Publishing Services (2012), S. 295–301

Kim, *Hokeun*, *Lee*, *Edward A.* (2017): Authentication and Authorization for the Internet of Things, in: IT Professional, 19 (2017), Nr. 5, S. 27–33

Köpf, *Boris*, *Basin*, *David* (2007): An Information-Theoretic Model for Adaptive Side-Channel Attacks, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, Alexandria, Virginia, USA: Association for Computing Machinery, 2007, S. 286–296

Krawczyk, *Dr. Hugo*, *Bellare*, *Mihir*, *Canetti*, *Ran* (1997), HMAC: Keyed-Hashing for Message Authentication, 2104, o. O., 1997-02, URL: https://www.rfc-editor.org/info/rfc2104

Krawczyk, *Hugo*, *Paterson*, *Kenneth G*, *Wee*, *Hoeteck* (2013): On the security of the TLS protocol: A systematic analysis, in: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I, Springer, o. O., 2013, S. 429–448

Krombholz, *Katharina*, *Mayer*, *Wilfried*, *Schmiedecker*, *Martin*, *Weippl*, *Edgar* (2017): "I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS, in: Proceedings of the 26th USENIX Security Symposium, Vancouver, BC, Canada, 2017-08, S. 1339–1356

Leiba, *Barry* (2012): OAuth Web Authorization Protocol, in: IEEE Internet Computing, 16 (2012), Nr. 1, S. 74–77

Morrissey, *P.*, *Smart*, *N. P.*, *Warinschi*, *B.* (2010): The TLS Handshake Protocol: A Modular Analysis, in: Journal of Cryptology, 23 (2010), Nr. 2, S. 187–223

Naylor, *David*, *Finamore*, *Alessandro*, *Leontiadis*, *Ilias*, *Grunenberger*, *Yan*, *Mellia*, *Marco*, *Maurizio*, *Munafò*, *Papagiannaki*, *Konstantina*, *Steenkiste*, *Peter* (2014): The Cost of the "S" in HTTPS, in: CoNEXT '14 (2014), S. 133–140

Oppliger, *Rolf*, *Hauser*, *Ralf*, *Basin*, *David* (2006): SSL/TLS session-aware user authentication – Or how to effectively thwart the man-in-the-middle, in: Computer Communications, 29 (2006), Nr. 12, S. 2238–2246

Paulson, *Lawrence C* (1999): Inductive analysis of the internet protocol TLS, in: ACM Transactions on Information and System Security (TISSEC), 2 (1999), Nr. 3, S. 332–351

Preneel, *Bart* (2003): Analysis and Design of Cryptographic Hash Functions, Englisch, thesis, Belgium: COSIC, 2003, 323 S., URL: https://www.e-reading.club/bookreader.php/141503/Analysis_and_Design_of_Cryptographic_Hash_Functions.pdf

Rosulek, *Mike* (2021): The Joy of Cryptography, Englisch, Corvallis, Oregon, USA, 2021-01-03, 286 S.

Shirey, *Dr. Rob* (2000), Internet Security Glossary, 2828, o. O., 2000-05, URL: https://www.rfc-editor.org/info/rfc2828

Sönmez Turan, *Meltem*, *Barker*, *Elaine*, *Burr*, *William*, *Chen*, *Lily* (2010), Reccomendation for Password-Based Key Derivation, Part 1: Storage Applications, SP 800-132, United States of America, 2010-12-22, URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

Sounthiraraj, *David*, *Sahs*, *Justin*, *Greenwood*, *Garret*, *Lin*, *Zhiqiang*, *Khan*, *Latifur* (2014): Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps, in: Network and Distributed System Security Symposium (NDSS). Internet Society, San Diego, CA, o. O., 2014, S. 1–14

Statista (2019): Welche Authentifizierungsverfahren nutzen Sie selber zum Schutz ihrer Daten und Geräte, Statista Umfrage Cybersecurity & Cloud 2018, statista, o. O.: statista, 2019-01-03, URL: https://de.statista.com/prognosen/952951/umfrage-in-deutschland-zu-authentifizierungsverfahren-zum-schutz-von-daten

Thielert, *Sandra* (2007): Kryptographische Algorithmen, Wernigerode: Rechenzentrum Hochschule Harz, 2007-05-11, URL: https://www.hs-harz.de/dokumente/extern/Rechenzentrum/Grundschutz/Kryptografische_Algorithmen.pdf

Zhang, *Liang*, *Choffnes*, *David*, *Levin*, *Dave*, *Tudor*, *Dumitraş*, *Mislove*, *Alan*, *Schulman*, *Aaron*, *Wilson*, *Christo* (2014): Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed, Englisch, in: Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, Vancouver, BC, Canada: Association for Computing Machinery, 2014-11-05, S. 489–502

Internetquellen

Cloudfare (o. J.): Warum HTTPS verwenden?, <https://www.cloudflare.com/de-de/learning/ssl/why-use-https/> (keine Datumsangabe) [Zugriff: 2023-05-28]

Encryption Consulting LLC (2023): What is SHA? What is SHA used for?, <https://www.encryptionconsulting.com/education-center/what-is-sha/> (2023-05) [Zugriff: 2023-05-21]

Krawisz, *Daniel* (2013): Chapter 2: Public-Key Cryptography, Crypto-Anarchy and Libertarian Entrepreneurship, <https://nakamotoinstitute.org/mempool/crypto-anarchy-and-libertarian-entrepreneurship-2/> (2013-05-24) [Zugriff: 2023-05-24]

MDN Web Docs (o. J.): SubtleCrypto: digest() method, Web APIs, <https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto/digest> (keine Datumsangabe) [Zugriff: 2023-06-08]

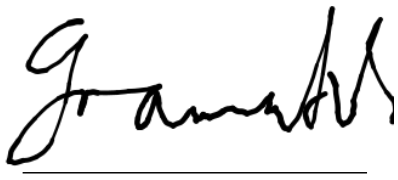
Okta (2023): Access Token: Definition, Architecture, Usage & More, <https://www.okta.com/identity-101/access-token/> (2023-02-14) [Zugriff: 2023-05-31]

Petitcolas, *Fabian* (o. J.): The information hiding homepage, Kerckhoffs's principles from « La cryptographie militaire », <https://www.petitcolas.net/kerckhoffs/index.html> (keine Datumsangabe) [Zugriff: 2023-04-14]

Webster, *Craig R.* (2009): Securing Passwords with Salt, Pepper and Rainbows, <http://www.barkingiguana.com/2009/08/03/securing-passwords-with-salt-pepper-and-rainbows/> (2009-08-03) [Zugriff: 2023-05-28]

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.



Ahaus, 21.6.2023

(Ort, Datum)

(Eigenhändige Unterschrift)