

Hybride Verschlüsselung und weiterführende Themen

Kryptologie IL (Vorlesungsteil)

Andreas Unterweger

Studiengang ITS
FH Salzburg

Sommersemester 2026

- Symmetrische Kryptografie
 - Schlüsseltausch
 - DH(E)
 - AES
- Asymmetrische Kryptografie
 - Öffentlicher Schlüssel
 - Privater/geheimer Schlüssel
 - RSA
- Kryptographische Hashes
 - Kollisionsresistenz
 - (Second-)Preimage-Resistenz
 - SHA-2

- Hybride Verschlüsselung
 - Schlüsseltausch wiederaufgegriffen (mit Schlüsselableitung)
 - Hybrides Verschlüsselungsschema
 - Challenge-Response-Schema
- Beispiele der kombinierten Verwendung von Kryptografie
 - Sichere Passwortspeicherung
 - Transport Layer Security
 - Praktische Protokolle (Auswahl)

- Diffie-Hellman-Schlüsseltausch
 - Alice und Bob potenzieren Zufallszahlen in einem Modul (multiplikative Gruppe)
 - Potenzregeln liefern Alice und Bob dasselbe Gruppenelement
 - Finales Gruppenelement kann verwendet werden, um einen gemeinsamen Schlüssel abzuleiten
 - Für Eve ist der gemeinsame Schlüssel unzumutbar zu berechnen (Diskretes-Logarithmus-Problem und Diffie-Hellman-Vermutungen)
- Herausforderungen
 - Sichere Gruppen wählen
 - Finales Gruppenelement in Schlüssel umwandeln → Schlüsselableitung

Schlüsseltausch wiederaufgegriffen II

- Schlüsselableitung

- Leite einen Schlüssel k aus dem finalen Gruppenelement f ab
- Beispiel: Erhalte 128-Bit-AES-Schlüssel aus 1.024-Bit-Gruppenelement
- Mehrere Möglichkeiten

- Hashen zur Schlüsselableitung

- Berechne kryptografischen Hash: $k := H(f)$
- Entfernt jede verbleibende algebraische Struktur
- Liefert ein Ergebnis fester Länge (Schlüssel)

- Standardisierte erweiterte Varianten, z.B. PBKDF2

- Wiederholte Verwendung von Hashes, z.B. 1.000 Iterationen
- Verlangsamt Angreifer deutlich
- Verwendung von zusätzlichem Salz (Details später)

[1] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. <https://www.ietf.org/rfc/rfc2898.txt> (Zugriff am 8.10.2022), 2000.

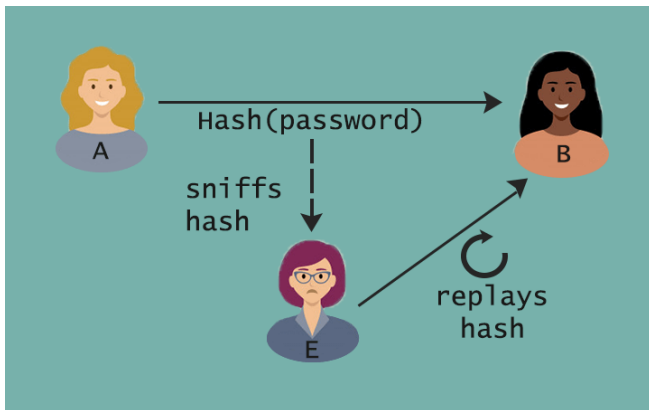
- Alternative zu Diffie-Hellman-Schlüsseltausch
- Kombiniere asymmetrische und symmetrische Chiffren
- Annahmen
 - Bob hat ein Schlüsselpaar ($k_{\text{oeffentlich}}^B, k_{\text{privat}}^B$)
 - Alice kennt Bobs öffentlichen Schlüssel (zur Erinnerung: Zertifikate!)
- Schlüsseltauschschritte:
 - 1 Alice generiert einen zufälligen Schlüssel k
 - 2 Alice verschlüsselt k mit Bobs öffentlichem Schlüssel:
 $k_e = E(k_{\text{oeffentlich}}^B, k)$
 - 3 Alice schickt den verschlüsselten Schlüssel k_e an Bob
 - 4 Bob entschlüsselt den empfangenen Schlüssel mit seinem geheimen Schlüssel: $k = D(k_{\text{privat}}^B, k_e)$
 - 5 Alice und Bob können Nachrichten mit dem symmetrischen Schlüssel k austauschen

Hybride Verschlüsselung II

- Warum nicht asymmetrische Kryptografie für alle Nachrichten verwenden?
 - Kryptografie mit öffentlichen Schlüsseln ist weit weniger effizient als Kryptografie mit geheimen Schlüsseln (längere Nachrichten benötigen viel aufwändigere Berechnungen)
 - RSA-Schlüssel sind viel länger als AES-Schlüssel mit demselben Sicherheitsniveau (zur Erinnerung: elliptische Kurven sind eine gangbare Alternative!)
- Zusammenfassung von hybrider Verschlüsselung
 - Verwende Kryptografie mit öffentlichen Schlüsseln, um den Schlüssel auszutauschen
 - Verwende Kryptografie mit geheimen Schlüsseln, um Nachrichten auszutauschen
 - Der symmetrische Schlüssel ist praktisch zufällig
 - Bobs öffentlicher Schlüssel muss vorab bekannt sein, damit das funktioniert
- Ergänzung zu diesem Schema: Challenge-Response-Schema

Challenge und Response I

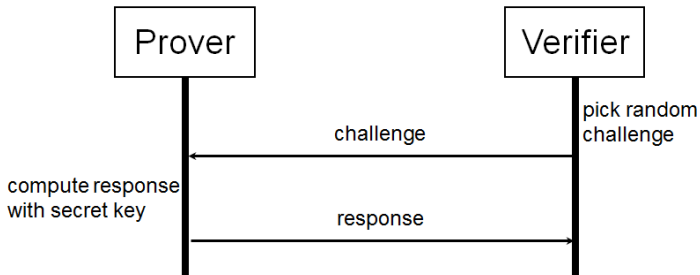
- Wie überprüfen, dass Bob mit Alice spricht und nicht mit Eve?
- Risiko von Replay-Angriffen durch Eve



Quelle: Gibson, D.: Replay Attacks. <https://cybersecurityglossary.com/replay-attacks/> (Zugriff am 8.10.2022), 2020.

Challenge und Response II

- Wie Authentifizierung lösen?
 - Zertifikate (fehlt es an Lebendigkeit, erlauben Replay-Angriffe)
 - Challenge und Response (verhindern Replay-Angriffe, indem sie Lebendigkeit prüfen)



Quelle: Rathgeb, C.: IT-Sicherheit, Kapitel 6 – Authentifikation.

https://www.dasec.h-da.de/wp-content/uploads/2014/05/06_authentifikation.pdf (Zugriff am 8.10.2022), 2014.

[2] Barrett, D. J. and Silverman, R. E.: SSH: The Secure Shell: The Definitive Guide, 1st ed., O'Reilly, 2001.

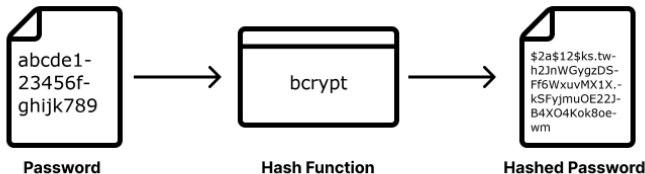
Challenge und Response III

- Beispiel: Challenge-Response-Schema zur Authentifizierung öffentlicher Schlüssel im Secure-Shell-(SSH)-Protokoll (vereinfacht):
 - 1 Alice sendet ihren öffentlichen Schlüssel $k_{\text{oeffentlich}}$ an Bob
 - 2 Bob generiert eine zufällige Challenge r und verschlüsselt sie mit Alices vermeintlichem öffentlichen Schlüssel: $c := E(k_{\text{oeffentlich}}, r)$
 - 3 Bob schickt die Challenge c an Alice
 - 4 Alice entschlüsselt die vermeintliche Challenge c' mit ihrem geheimen Schlüssel k_{geheim} : $r' = D(k_{\text{geheim}}, c')$
 - 5 Alice hasht die entschlüsselte Challenge r' : $h' := H(r')$
 - 6 Alice schickt die gehashte Antwort (Response) h' an Bob
 - 7 Bob hasht seine zufällige Challenge r : $h := H(r)$
 - 8 Bob überprüft, ob $h = h'$ (wenn nicht, könnte er mit Eve sprechen)
- Zufälligkeit verhindert Replay-Angriffe
- Hashing ist erforderlich, damit Chosen-Plaintext-Angriffe vermieden werden können

Überblick über sichere Passwortspeicherung

- Wie (Benutzer)-Passwörter speichern?
 - Nicht im Klartext (offensichtlich)
 - Verschlüsselt? → Schlüssel zur Entschlüsselung liefert Klartextpasswörter
 - Gehasht? → Identische Passwörter liefern identische Hashes

Password Hashing

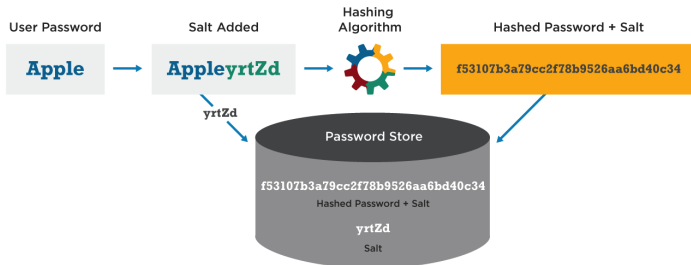


Quelle: Authgear: Password Hashing and Salting Explained. <https://www.authgear.com/post/password-hashing-salting> (Zugriff am 8.10.2022), 2022.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords. <https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (Zugriff am 8.10.2022), 2021.

- Salz: Zufallszahl (zusammen mit Passwort gespeichert) **pro Benutzer**
- Salzen: Passwort p und Salz s vor dem Hashen aneinanderhängen
- Gesalzenes Hashing: $\hat{H}(p, s) := H(p||s)$

Password Hash Salting



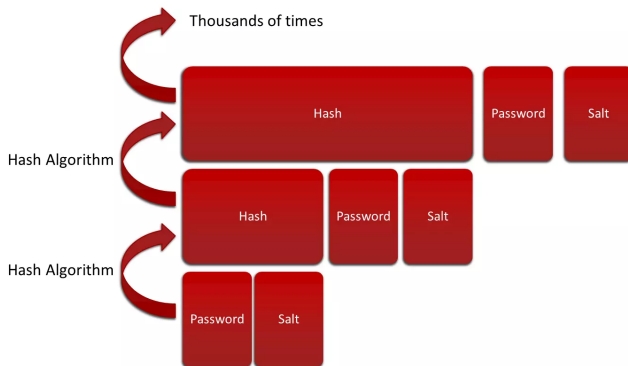
Quelle: Taylor, C.: Password Salting. <https://cyberhoot.com/cybrary/password-salting/> (Zugriff am 8.10.2022), 2020.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (Zugriff am 8.10.2022), 2021.

Strecken

- Strecken: Berechnung durch wiederholtes Hashen verlängern
- Iterationen: $P_1 = \hat{H}(p, s)$, $P_{n>1} = \hat{H}(P_{n-1} || p, s)$
- Gespeicherter Passworthash: P_N für große N , z.B. 10.000



Quelle: Romero, M. I.: Risks and Challenges of Password Hashing.

<https://www.sitepoint.com/risks-challenges-password-hashing/> (Zugriff am 8.10.2022), 2014.

Angriffe auf sicher gespeicherte Passwörter I

- Rainbow Tables (vorberechnete Hashes, wenn kein Salz verwendet wird)
- Wörterbuchangriff (versuche eine Liste von üblichen Passwörtern)
- Brute Force (versuche alle Passwörter aus einer definierten Menge)

				
Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	-	-	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	1vn49sa	z32i6t0

Adaptiert aus Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (Zugriff am 8.10.2022), 2021.

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (Zugriff am 8.10.2022), 2021.

Angriffe auf sicher gespeicherte Passwörter II

- Brute-Force-Angriffe hängen vom angen. Passwortalphabet ab
 - Beispiel: 8-Zeichen-Passwörter mit englischem Alphabet (nur Groß- und Kleinbuchstaben): Schlüsselraum ist
$$52^8 > 50^8 = \left(\frac{100}{2}\right)^8 = \frac{(10^2)^8}{2^8} = \frac{10^{16}}{2^8} = 2^2 \cdot \frac{10^{16}}{2^{10}} \approx 4 \cdot \frac{10^{16}}{10^3} = 4 \cdot 10^{13}$$
 - Wörterbuchangriffe sind nur erfolgreich, wenn das Passwort im Wörterbuch enthalten ist (andernfalls Brute-Force-Angriff erforderlich)
 - Strecken kann Wörterbuch- **und** Brute-Force-Angriffe verlangsamen
 - Salzen macht Rainbow Tables zwecklos
- Salzen und Strecken verwenden (besser: standardisierte Algorithmen)
- Benutzerempfehlung: Verwende einen Passwortmanager

[3] Arias, D.: Adding Salt to Hashing: A Better Way to Store Passwords.

<https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/> (Zugriff am 8.10.2022), 2021.

Überblick über Transport Layer Security (TLS)

- Basiert auf inzwischen veraltetem Secure Sockets Layer (SSL)
- Protokoll, um sicheren Kanal zwischen Alice und Bob bereitzustellen
- Garantien
 - Authentifizierung (Bob ist immer authentifiziert, Alice optional)
 - Vertraulichkeit (nur Alice und Bob können Daten lesen, ein Angreifer aber nicht)
 - Integrität (Daten können nicht unerkant von einem Angreifer verändert werden)
- Verwendet eine Vielfalt an kryptografischen Primitiven, z.B.
 - Asymmetrische Chiffren
 - Symmetrische Chiffren
 - Kryptografische Hashes
 - Digitale Signaturen
 - Digitale Zertifikate

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

- Vom Unterzeichner/Sender generiert und an die Nachricht angehängt
- Kann vom Empfänger (oder jedem anderen) überprüft werden
- Bewahrt die Nachrichtenintegrität anstatt der -privatsphäre
- Garantien
 - Die signierte Nachricht wurde nicht verändert
 - Die signierte Nachricht stammt vom Unterzeichner
- Nichtabstreitbarkeit: Der Unterzeichner kann nicht behaupten, die Nachricht **nicht** unterzeichnet zu haben
- Basieren auf Kryptografie mit öffentlichen Schlüsseln und Hashes
 - Signatur: Hash der Nachricht, verschlüsselt mit dem geheimen Schlüssel
 - Signaturüberprüfung: Vergleiche Hash und entschlüsselte Signatur
 - Wenn die Signaturüberprüfung fehlschlägt, ist entweder die Nachricht verfälscht worden oder jemand anderer (ein anderer Schlüssel) hat die Nachricht signiert

- Digitale Zertifikate

- assoziieren einen öffentlichen Schlüssel mit einer Person/Entität
- enthalten Daten zum Unterzeichner, der Person und zum Schlüssel
- ist von einer vertrauenswürdigen Partei signiert
- erlaubt die Überprüfung der Signatur und dadurch Identität-zu-Schlüssel-Übereinstimmung
- Annahme: Vertrauenswürdige Partei, deren öffentlicher Schlüssel allen Parteien bekannt ist

- Zertifizierungsstellen (CAs)

- wird vertraut, Zertifikate auszustellen
- überprüfen die Identität des Besitzers oder der Entität
- haben angenommenerweise einen bekannten öffentlichen Schlüssel

- Zertifizierungsstellenhierarchien (CA-Hierarchien)

- Ausstellen von Zertifikaten ist an Sub-CA(s) delegiert
- Mehrstufige Zertifikate von der CA zu(r) Sub-CA(s) zur Entität

→ Hierarchisches Vertrauensmodell

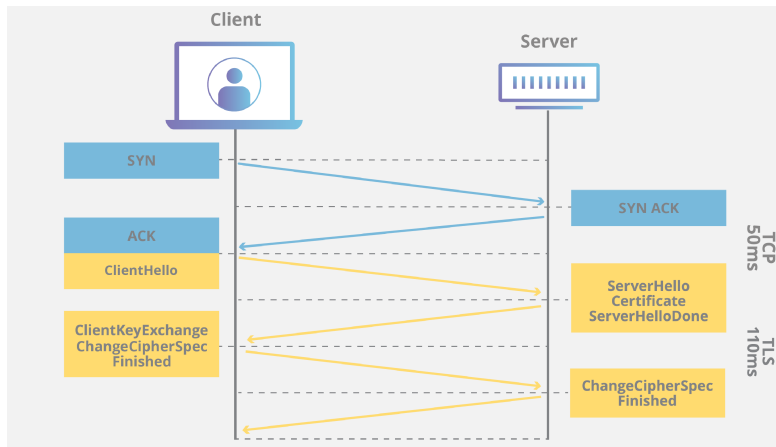
- Stammzertifizierungsstelle(n) signieren ihr(e) eigenes/n Zertifikat(e) selbst

Schlüssel- und Chiffre-Vereinbarung I

- TLS baut auf Transportprotokolle wie TCP auf
- Initialer Handshake
 - Authentifiziert kommunizierende Parteien durch Zertifikate
 - Verhandelt verwendete Chiffren (nicht jede Partei unterstützt jede Chiffre)
 - Führt Schlüsseltausch für Forward Secrecy durch
 - Manipulationssicher sogar gegen aktive Angreifer (die Nachrichten verändern)
- Verwendete kryptografische Primitive
 - Digitale Zertifikate zur Authentifizierung (vereinfacht)
 - Digitale Signaturen für unterstützte Chiffren und Zertifikate
 - (EC)DHE and ähnliche Protokolle zum Schlüsseltausch
 - Hashes der Abschrift (Abfolge von Handshake-Nachrichten)

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

Schlüssel- und Chiffre-Vereinbarung II



Quelle: Cloudflare: What is Transport Layer Security (TLS)?

<https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/> (Zugriff am 26.10.2022), 2022.

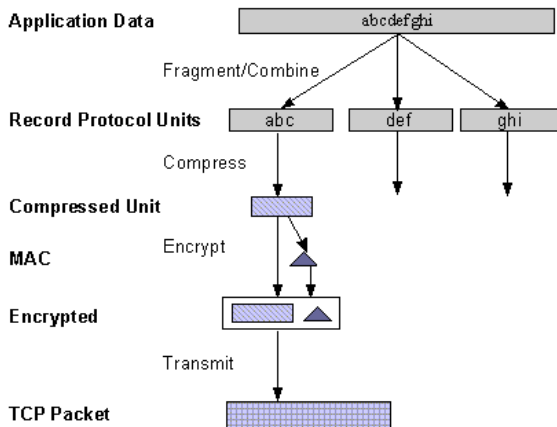
[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).

<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

- TLS kann Nachrichten nach beendetem Handshake austauschen
- Record-Protokoll
 - ① (Alice) Nachricht wird in Blöcke aufgespalten
 - ② (Alice) Blöcke (Records) werden geschützt
 - ③ (Alice) Geschützte Records werden übertragen
 - ④ (Bob) Geschützte Records werden empfangen
 - ⑤ (Bob) Geschützte Records werden überprüft und der Schutz entfernt
 - ⑥ (Bob) Blöcke (Records) werden wieder zu einer Nachricht zusammengesetzt
- Verwendete kryptografische Primitive
 - Symmetrische Block-Chiffren in ausgewählten Modi (ohne Details)
 - Message-Authentication-Codes (MACs, ähnlich wie Hashes, ohne Details)
- Optionaler Kompressionsschritt (ohne Details)

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

Sicherer Nachrichtenaustausch II



Quelle: The Apache Software Foundation: SSL/TLS Strong Encryption: An Introduction.
https://httpd.apache.org/docs/2.4/ssl/ssl_intro.html (Zugriff am 26.10.2022), 2022.

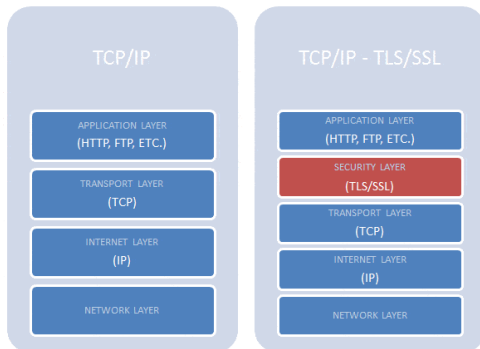
[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

- Neuere Versionen von TLS implementieren neue Chiffren und Features
 - SSL und ältere TLS-Versionen sind
 - verwundbar gegenüber bestimmten Angriffen (ohne Details)
 - unterstützen gebrochene Chiffren/Hashalgorithmen etc.
 - unterstützen schwache Algorithmen und Schlüssellängen
- Abkündigung (engl. *deprecation*) oder Entfernung von mangelhaften/schwachen Algorithmen und Protokollteilen
- Versionierung (aktuell: TLS 1.3)
- Byte-für-Byte-Illustration einer TLS-1.3-Verbindung (inklusive Handshake): <https://tls13.xargs.org/>

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).
<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

Praktische Belange II

- TLS ist ausgelegt, um eine zusätzliche Schicht für Anwendungs(-niveau-)protokolle bereitzustellen



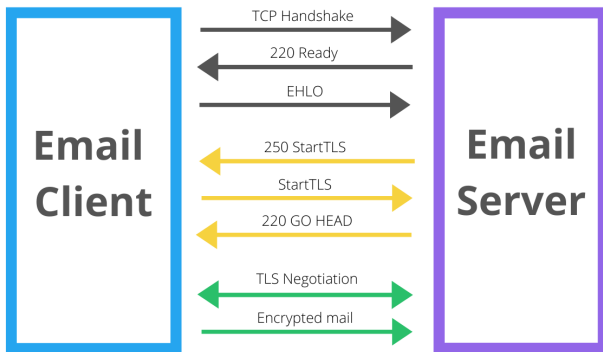
Quelle: Slaviero, M.: TLS/SSL and .NET Framework 4.0.

<https://www.red-gate.com/simple-talk/development/dotnet-development/tlsssl-and-net-framework-4-0/> (Zugriff am 26.10.2022), 2011.

[4] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446).

<https://www.rfc-editor.org/rfc/rfc8446.html> (Zugriff am 26.10.2022), 2018.

- Nachrüstung ungeschützter Protokolle durch STARTTLS-Kommando



Quelle: Griffin, J.: What is StartTLS? <https://sendgrid.com/blog/what-is-starttls/> (Zugriff am 26.10.2022), 2020.

[5] Griffin, J.: What is StartTLS? <https://sendgrid.com/blog/what-is-starttls/> (Zugriff am 26.10.2022), 2020.

Praktische Protokolle (Auswahl)

- Praktische kryptografische Protokolle zur Illustration der kombinierten Verwendung von Kryptografie (wähle eines):
 - Masking (Smart-Meter-Datenaggregation) [6]
 - Proof-of-Work-Blockchains (elektronisches Geld) [7]
 - Wi-Fi Protected Access 3 (WPA3, drahtlose Kommunikation) [8, 9]
 - Inhaltsschutz (Videostreaming) [10, 11]
 - Weitere Themen (Vorschläge)

[6] Kursawe, K., Danezis, G. and Kohlweiss, M.: Privacy-friendly Aggregation for the Smart Grid. In PETS 2011: Privacy Enhanced Technology Symposium, pp. 175–191, 2011.

[7] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. <https://bitcoin.org/bitcoin.pdf> (Zugriff am 26.10.2022), 2008.

[8] WiFi Alliance: WPA3™ Specification Version 3.3. <https://www.wi-fi.org/file/wpa3-specification> (Zugriff am 10.7.2024), 2024.

[9] WiFi Alliance: Wi-Fi Protected Access® Security Considerations.

https://www.wi-fi.org/system/files/Security_Considerations_20210511.pdf (Zugriff am 10.7.2024), 2021.

[10] European Telecommunications Standards Institute and European Broadcasting Union: Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 2: CPCM Reference Model (ETSI TS 102 825-2 V1.2.1), 2011.

[11] European Telecommunications Standards Institute and European Broadcasting Union: Digital Video Broadcasting (DVB); Content Protection and Copy Management (DVB-CPCM); Part 5: CPCM Security Toolbox (ETSI TS 102 825-5 V1.2.1), 2011.

Danke für die Aufmerksamkeit!

Fragen?