IT-Sicherheit 4 (Signaturen & Zertifikate) – Zusammenfassung mit IHK-Bezug

Hashfunktionen, digitale Signatur, Zertifikate (X.509), PKI (RA · CA · VA), Signaturstufen nach eIDAS

✍️ IT-Sicherheit 4 – Signaturen & Zertifikate

📘 Grundlage: Folien „IT-Sicherheit: Signaturen & Zertifikate“ (GBS Fachschule, 30 Folien).

🧭 So liest du dieses Dokument

- **Fett** = Prüfungs-Schlagwort. 🎯 = eigene Einschätzung der IHK-Relevanz.
- Dieses Kapitel ist **Kernstoff**. Hashwerte, Signaturen und Zertifikate kommen in **AP1** (Schutzziele, Integrität) und in **AP2** (HTTPS, Zertifikate, PKI, E-Mail-Signatur) regelmäßig vor.
- ✏️ Übungsfragen am Ende jedes Blocks, Lösungen ganz am Ende. 🔎 = Ergänzung (nicht auf den Folien). ⚠️ = Falle oder Hinweis zu einer Folie.

#️⃣ Block 1: Hashfunktionen

🎯 IHK-Relevanz: **sehr hoch**.

Eine **Hashfunktion** bildet **beliebig große** Daten auf einen Wert **fester Länge** ab.

- Das Ergebnis heißt **Hashwert**, **Prüfsumme** oder **Message Digest**.
- Er ist der **„digitale Fingerabdruck“** einer Datei.

Prinzip-Beispiel der Folie (Quersumme bis zur Einstelligkeit):

Eingabe	Rechnung	Hash
132	1 + 3 + 2	**6**
56	5 + 6 = 11 → 1 + 1	**2**
8202	8 + 2 + 0 + 2 = 12 → 1 + 2	**3**
52572	5 + 2 + 5 + 7 + 2 = 21 → 2 + 1	**3**

⚠️ Das ist **keine brauchbare** Hashfunktion. Sie zeigt nur das Prinzip. 8202 und 52572 ergeben beide 3, das ist schon eine **Kollision**.

Anforderungen an eine kryptographische Hashfunktion

Eigenschaft	Bedeutung	Quersummen-Beispiel
Einwegfunktion	vom Hashwert kommt man **nicht** zurück zur Eingabe	✔️ erfüllt
Kollisionsresistenz	zwei verschiedene Eingaben ergeben **praktisch nie** denselben Hash	✘ nur 10 mögliche Werte (0–9) → spätestens der 11. Wert kollidiert
Lawineneffekt	winzige Änderung der Eingabe → **komplett anderer** Hash	✘
🔎 **feste Länge**	egal wie groß die Eingabe ist	✔️
🔎 **deterministisch**	gleiche Eingabe → immer gleicher Hash	✔️

Warum es Kollisionen immer geben muss: Es gibt unendlich viele mögliche Eingaben, aber nur endlich viele Hashwerte. Die Lösung ist, **sehr viele** Hashwerte zu ermöglichen. Mit 256 Bit gibt es $\mathbf{2^{256} \approx 1{,}16 \cdot 10^{77}}$ Möglichkeiten, also eine Zahl mit 78 Stellen. Eine zufällige Kollision ist damit praktisch ausgeschlossen.

💡 Merksatz: **Eine Hashfunktion gilt als sicher, solange niemand eine Kollision gezielt erzeugen kann.**

Verfahren

Verfahren	Länge	Status
MD5	128 Bit	❌ **unsicher**, Kollisionen nachgewiesen
SHA-1	160 Bit	❌ **unsicher**, Kollisionen nachgewiesen (🔎 2017 „SHAttered“)
SHA-256 / SHA-512 (SHA-2)	256 / 512 Bit	✅ **Standard**, z. B. bei Zertifikaten und Downloads
🔎 **SHA-3**	224–512 Bit	✅ neuer Standard, anderes Bauprinzip
RIPEMD-160,	160 / 512 Bit	selten

Verfahren	Länge	Status
Whirlpool		

🔎 Für **Passwörter** nimmt man **langsame** Verfahren wie **bcrypt**, **Argon2** oder **yescrypt** (unter Debian y in /etc/shadow). SHA-256 ist dafür zu schnell, ein Angreifer könnte Milliarden Versuche pro Sekunde rechnen.

Notation und Lawineneffekt

Hashwerte schreibt man **hexadezimal**. 4 Bit ergeben 1 Hex-Ziffer:

1111 1010 1011 0110 1101 0010 1011 1110 → f a b 6 d 2 b e → **fab6d2be**

Eingabe	SHA-1
Hallo Welt	28cbbc72d6a52617a7abbfff6756d04bbad0106a
Hallo Welt**!**	726c3e8861ab0652a5043ea5faff6d3ef33fb209
Hallo und guten Tag! Wie geht‘s?	aedf6fce7d244cedb9e1a377aff3dd4da7e80585

- Ein einziges „!“ ändert den **ganzen** Hash. Das ist der **Lawineneffekt**.
- Ein langer Text ergibt einen **gleich langen** Hash: SHA-1 hat immer 40 Hex-Ziffern = 160 Bit.
- 🔎 Werte nachgeprüft. Achtung: Mit einem Zeilenumbruch am Ende (z. B. echo "Hallo Welt" | sha1sum) kommt ein anderer Wert heraus. Ohne Umbruch geht es mit printf 'Hallo Welt' | sha1sum.

Einsatz

Einsatz	Wie?
Datenübertragung / Downloads prüfen	Hash an Quelle und Ziel bilden und vergleichen. Gleich → Übertragung in Ordnung. 🔎 sha256sum debian.iso, unter Windows Get-FileHash
Dubletten finden	Hashes vergleichen ist viel schneller als ganze Dateien
Passwörter speichern	nur der Hash wird gespeichert. Beim Login wird die Eingabe gehasht und verglichen. 🔎 mit **Salt**

🔎 **Salt** = zufälliger, **nicht geheimer** Zusatzwert pro Benutzer, der vor dem Hashen an das Passwort gehängt wird. Gleiche Passwörter ergeben so verschiedene Hashes. **Rainbow Tables** (vorberechnete Hash-Tabellen) werden nutzlos, und jedes Konto muss einzeln geknackt werden. Gegen schwache Passwörter hilft Salt aber **nicht**.

⚠️ Hashen ist keine Verschlüsselung

	Verschlüsselung	Hashfunktion
Schlüssel	ja	**nein**
umkehrbar	ja (mit Schlüssel)	**nein** (Einwegfunktion)
Ausgabelänge	wächst mit der Eingabe	**immer gleich**
Schutzziel	**Vertraulichkeit**	**Integrität**

✏️ Übungsfragen Block 1

Ü1.1 Nennen Sie zwei Anforderungen an eine kryptographische Hashfunktion und erklären Sie sie kurz.

Ü1.2 Ein Kollege sagt: „Unsere Passwörter sind sicher, die sind ja mit MD5 verschlüsselt.“ Nehmen Sie Stellung (zwei Fehler).

Ü1.3 Warum kann keine Hashfunktion Kollisionen vollständig ausschließen?

✍️ Block 2: Digitale Signatur

🎯 IHK-Relevanz: **sehr hoch**.

Manipulationen (absichtlich, versehentlich oder durch Übertragungsfehler) lassen sich **nicht verhindern**. Eine digitale Signatur macht sie aber **erkennbar**.

Schutzziele: Integrität (Daten unverändert) und 🔎 **Authentizität** (stammt wirklich vom Absender). 🔎 Daraus folgt auch **Nichtabstreitbarkeit**: Der Absender kann später nicht leugnen, signiert zu haben.

Erster Versuch: Datei + Hash mitschicken

1. Der Absender bildet den Hash der Datei.
2. Er schickt Datei, Hash und den Namen der Hashfunktion.
3. Der Empfänger hasht die Datei selbst und vergleicht.

Der Haken: Ein Angreifer fängt alles ab, ändert die Datei, **berechnet einfach einen neuen Hash** und schickt beides weiter. Der Empfänger merkt nichts. → Der **Original-Hash muss geschützt werden**. Dafür braucht man die **asymmetrische Kryptographie**.

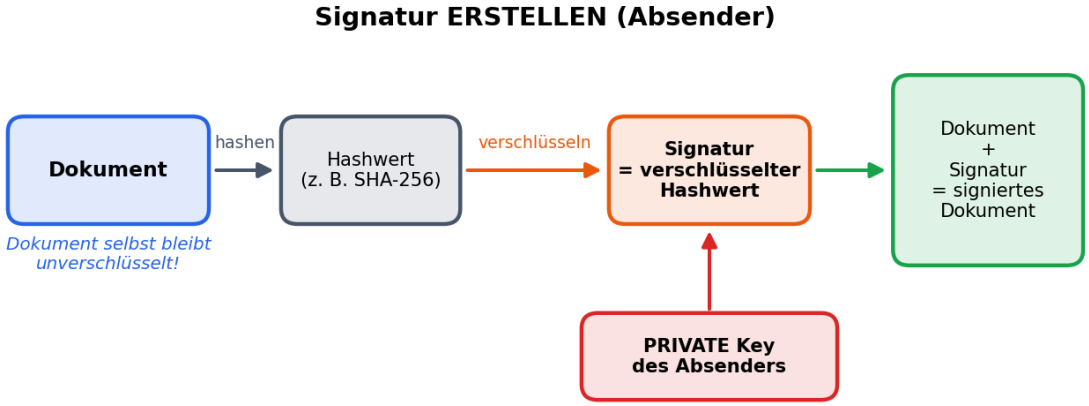
Asymmetrie „umgedreht“

Verschlüsseln mit …	Entschlüsseln mit …	Zweck
Public Key des Empfängers	Private Key des Empfängers	**Vertraulichkeit** (Verschlüsselung, ITS 3)

Verschlüsseln mit …	Entschlüsseln mit …	Zweck
Private Key des Absenders	Public Key des Absenders	**Signatur**: jeder kann prüfen, nur der Absender konnte es erzeugen

💡 Merksatz: **Verschlüsseln → Public Key des Empfängers. Signieren → eigener Private Key.**

Signatur erstellen

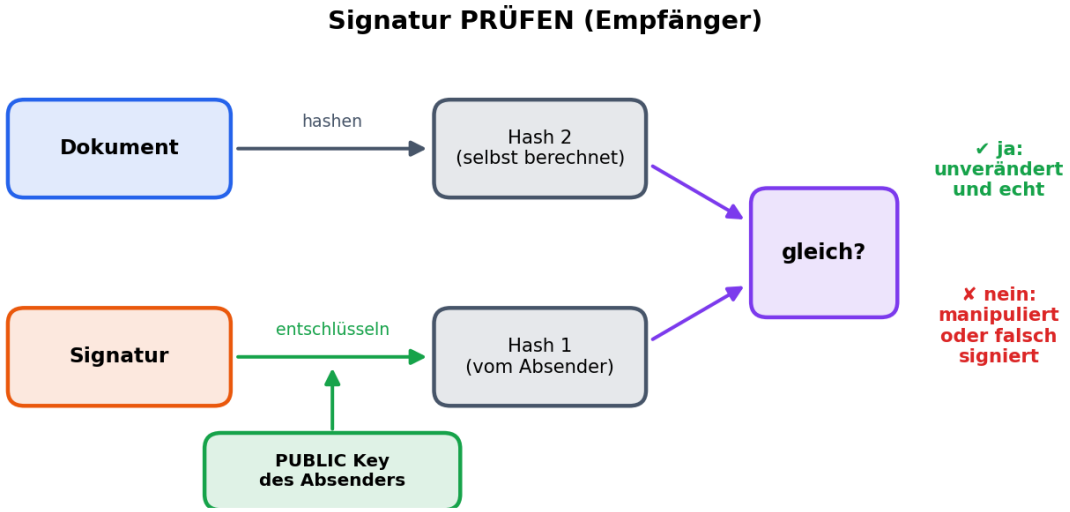


Signatur erstellen

1. Den **Hash** des Dokuments bilden.
2. Den Hash mit dem **eigenen Private Key** verschlüsseln. Das Ergebnis ist die **Signatur**.
3. Dokument und Signatur zusammen verschicken. Das Dokument selbst bleibt **unverschlüsselt**.

Begriff der Folie	Bedeutung
Signaturfunktion	die benutzte Hashfunktion
Signaturschlüssel	der Private Key des Signierenden
Signatur	der verschlüsselte Hashwert

Signatur prüfen



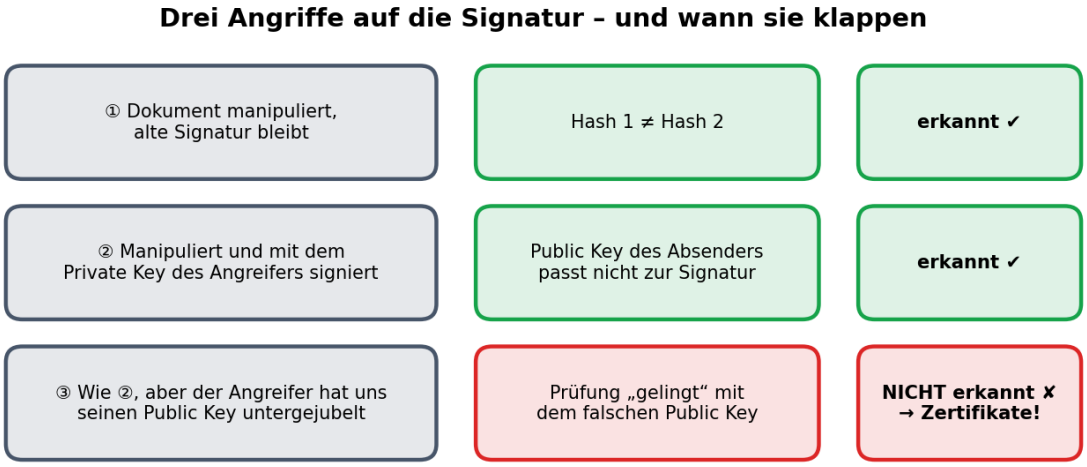
Signatur prüfen

1. Die Signatur mit dem **Public Key des Absenders** entschlüsseln → **Hash 1** (Original).
2. Das empfangene Dokument selbst hashen → **Hash 2**.
3. Vergleichen: **gleich** → Dokument unverändert und vom Absender. **Ungleich** → manipuliert.

⚠️ Eine Signatur macht das Dokument **nicht vertraulich**. Jeder kann es lesen. Wer beides will, signiert **und** verschlüsselt (🔎 z. B. S/MIME oder PGP bei E-Mail).

🔎 Das „Verschlüsseln mit dem Private Key“ ist das Schulmodell und passt genau für **RSA**. Andere Verfahren wie **ECDSA** signieren mathematisch anders. Das Prinzip „mit Private Key signieren, mit Public Key prüfen“ gilt aber überall.

Angriffe



Angriffe

Angriff	Ergebnis
① Dokument geändert, alte Signatur bleibt	Hash 1 ≠ Hash 2 → **erkannt**
② geändert und mit dem **Private Key des Angreifers** signiert	Mit dem echten Public Key des Absenders lässt sich die Signatur nicht sinnvoll entschlüsseln → **erkannt**
③ wie ②, und der Angreifer hat uns **seinen Public Key als den des Absenders untergejubelt**	Prüfung „gelingt“ → **nicht erkannt** ❌

💡 Das zentrale Problem: **Woher weiß ich, dass ein Public Key wirklich zu der Person oder dem Server gehört? → Zertifikate**.

✏️ Übungsfragen Block 2

Ü2.1 Warum reicht es nicht, eine Datei einfach zusammen mit ihrem Hashwert zu verschicken?

Ü2.2 Anna will Ben ein Dokument signiert schicken. Mit welchem Schlüssel signiert Anna, mit welchem prüft Ben?

Ü2.3 Welche Schutzziele erfüllt eine digitale Signatur, welches nicht?

📜 Block 3: Digitale Zertifikate

🎯 IHK-Relevanz: **sehr hoch** – HTTPS und Zertifikate sind AP2-Klassiker.

Ein **Zertifikat** ist im Kern ein **Public Key, der von einer vertrauenswürdigen Stelle signiert** wurde. Es bestätigt: „Dieser Public Key gehört wirklich zu shop.com.“ Schutzziel: **Authentizität**.

⚠️ Das lässt sich **nicht rein kryptographisch** lösen. Am Ende muss man **jemandem vertrauen**.

Bestandteile nach X.509

Feld	Frage
Aussteller (Zertifizierungsstelle, CA)	Wer hat zertifiziert?
Inhaber (Subject, z. B. shop.com)	Was wurde zertifiziert?
Public Key des Inhabers	um den geht es
Signaturalgorithmus (z. B. sha256WithRSAEncryption)	Wie wurde signiert?
Gültigkeitsdauer (gültig von / bis)	Wie lange gilt es?
Seriennummer	eindeutige Nummer, wichtig zum **Sperren**
… und die **Signatur der CA** über alle diese Angaben	

🔎 **Laufzeit von Webserver-Zertifikaten:** Auf der Folie stehen 1–2 Jahre. Das ist veraltet. Seit März 2026 sind höchstens **200 Tage** erlaubt. Ab 2027 sinkt die Grenze auf 100 Tage, ab 2029 auf **47 Tage**. Deshalb erneuert man Zertifikate heute **automatisch** (z. B. **Let’s Encrypt** mit **ACME**/certbot, Laufzeit 90 Tage).

Zertifikatsprüfung = Signaturprüfung

1. Die Signatur im Zertifikat mit dem **Public Key der CA** entschlüsseln → Original-Hash.
2. Die Zertifikatsdaten selbst hashen.
3. Vergleichen. Gleich → Zertifikat und damit der Public Key sind **echt**.
4. 🔎 Außerdem prüft der Browser: Ist es **noch gültig**, passt der **Name** zur aufgerufenen Adresse, wurde es **gesperrt**?

Wem vertraut man? Die „vertrauenswürdigen Stellen“

- Um die CA-Signatur zu prüfen, braucht man den **Public Key der CA**. Wer garantiert dessen Echtheit?
- Lösung: Betriebssystem und Browser bringen ab Werk eine Liste von **Root-Zertifikaten** mit (🔎 **Zertifikatsspeicher / Root Store**, unter Windows certmgr.msc, unter Debian /etc/ssl/certs).
- **Wer** darin steht, entscheiden die **Hersteller** (Microsoft, Apple, Google, Mozilla …). Wir können dem nur vertrauen.

Ergänzung: Vertrauenskette (Chain of Trust)



Der Browser prüft von rechts nach links, bis er bei einer Root-CA ankommt, der er vertraut

Vertrauenskette

🔎 **Zertifikatskette (Chain of Trust):** Die Root-CA signiert meist nicht direkt, sondern eine **Zwischen-CA** (Intermediate), und diese signiert das Server-Zertifikat. Der Browser prüft die Kette, bis er bei einer Root-CA ankommt, der er vertraut.

🔎 **Selbstsigniertes Zertifikat:** Aussteller = Inhaber, keine CA hat bestätigt. Der Browser warnt („Verbindung ist nicht privat“). Das ist in Ordnung für interne Tests oder Laborserver, aber nicht für öffentliche Seiten.

✏️ Übungsfragen Block 3

Ü3.1 Was ist ein digitales Zertifikat? Antworten Sie in einem Satz.

Ü3.2 Nennen Sie vier Angaben, die ein X.509-Zertifikat enthält.

Ü3.3 Ein Kunde ruft das Firmen-Intranet auf und bekommt die Warnung „Zertifikat nicht vertrauenswürdig“. Nennen Sie zwei mögliche Ursachen.

🏛️ Block 4: PKI – Public Key Infrastructure

🎯 IHK-Relevanz: **hoch** – die Abkürzungen RA, CA und VA werden gern abgefragt.

PKI = ein System, das Zertifikate **ausstellt, verwaltet und prüft**.

Stelle	Name	Aufgabe
RA	Registration Authority – **Registrierungsstelle**	nimmt Anträge an, **prüft die Identität** des Antragstellers, gibt sie an die CA weiter
CA	Certificate Authority – **Zertifizierungsstelle**	**stellt das Zertifikat aus** (signiert es) und informiert die VA
VA	Validation Authority –	beantwortet Anfragen:

Stelle	Name	Aufgabe
	Validierungsdienst	**Ist dieses Zertifikat gültig?**



PKI-Ablauf

Ablauf am Beispiel shop.com:

1. Der Betreiber beantragt ein Zertifikat bei der **RA**.
2. Die RA prüft den Antrag und gibt ihn an die **CA** weiter.
3. Die CA stellt das Zertifikat aus …
4. … und informiert die **VA**.
5. Der Betreiber hinterlegt das Zertifikat auf seinem **Webserver**.
6. Ruft ein Kunde shop.com auf, fragt der **Browser** bei der VA nach, ob das Zertifikat gültig ist.

🔎 Gesperrte Zertifikate (z. B. weil der Private Key gestohlen wurde) stehen in einer **Sperrliste (CRL, Certificate Revocation List)**. Online abfragen kann man sie per **OCSP** (Online Certificate Status Protocol).

✏️ Übungsfragen Block 4

Ü4.1 Wofür stehen RA, CA und VA, und was macht jede Stelle?

Ü4.2 Der Private Key eines Webservers wurde gestohlen. Was muss mit dem Zertifikat passieren, und wie erfahren Browser davon?

⚖️ Block 5: Elektronische Signaturen im Recht (eIDAS)

🎯 IHK-Relevanz: **mittel** – die drei Stufen kommen vor, vor allem die **QES**.

🔎 Rechtsgrundlage ist die EU-Verordnung **eIDAS** (gilt in allen EU-Staaten).

Stufe	Sicherheit	Technik	Beispiele	vor Gericht
Einfache elektronische Signatur	sehr niedrig	**keine** Kryptographie	Name unter einer Mail, eingescannte Unterschrift, Unterschrift auf dem Paket-Lesegerät	einsetzbar, aber leicht zu bestreiten. Das Gericht entscheidet frei
Fortgeschrittene elektronische Signatur	hoch	**digitales Zertifikat**. Der Aussteller ist egal, auch ein **selbst erstelltes** Zertifikat geht	Signatur-Datei im Anhang (🔎 z. B. S/MIME, PGP)	einsetzbar, Echtheit muss ggf. **bewiesen** werden
Qualifizierte elektronische Signatur (**QES**)	sehr hoch	technisch wie die fortgeschrittene, aber mit Zusatzanforderungen (siehe unten)	Vertrag, Bescheid, beA	**gleichgestellt mit der handschriftlichen Unterschrift** (EU-weit)

Zusätzlich bei der QES:

- Das Zertifikat gibt es erst nach einer **Identitätsprüfung** (z. B. Personalausweis).
- Es muss ein **qualifiziertes Zertifikat** eines **anerkannten** Vertrauensdienstes sein. **Keine** selbst erstellten Zertifikate.
- Signiert wird mit einer **sicheren Signaturerstellungseinheit (SSEE)**, z. B. Chipkarte + Kartenleser.
- Die SSEE ist von einer anerkannten Stelle **geprüft**.
- Der **Private Key** liegt **nur in der SSEE** und kann nicht herauskopiert werden.

🔎 Heute gibt es auch die **Fernsignatur**: Der Private Key liegt in einem zertifizierten Hochsicherheitsmodul (HSM) beim Anbieter, und man löst die Signatur per App oder TAN aus. Das ist ebenfalls eine QES.

Anerkannte Anbieter in Deutschland (Folie): D-Trust (Bundesdruckerei), Bundesnotarkammer, Signtrust (Deutsche Post), S-Trust (Deutscher Sparkassenverlag), Telesec (Deutsche Telekom). Die EU-weite Liste steht unter *eidas.ec.europa.eu*.

💡 Merksatz: **Nur die QES ersetzt die Unterschrift auf Papier.**

🔎 Bezug zu **Recht**: Wo das Gesetz die **Schriftform** verlangt, kann sie meist durch die QES ersetzt werden (§ 126a BGB). Ausnahme: Die **Kündigung eines Arbeitsverhältnisses** muss weiterhin auf Papier eigenhändig unterschrieben sein (§ 623 BGB). Eine E-Mail oder QES reicht dort **nicht**.

✏️ Übungsfragen Block 5

Ü5.1 Ihr Betrieb will Verträge mit Kunden rein digital unterschreiben lassen, rechtlich so sicher wie auf Papier. Welche Signaturstufe brauchen Sie, und was ist dafür nötig?

Ü5.2 Warum ist eine Signatur mit einem selbst erstellten Zertifikat keine QES?

📋 Spickzettel

Begriff	In einem Satz
Hashfunktion	beliebige Daten → Fingerabdruck fester Länge, nicht umkehrbar
Kollisionsresistenz	zwei Eingaben ergeben praktisch nie denselben Hash
Lawineneffekt	kleine Änderung → völlig anderer Hash
MD5 / SHA-1	veraltet, Kollisionen bekannt
SHA-256	aktueller Standard, 256 Bit
Salt	zufälliger Zusatz pro Passwort, nicht geheim, gegen Rainbow Tables
Digitale Signatur	Hash, verschlüsselt mit dem eigenen Private Key → Integrität + Authentizität
Signatur prüfen	mit dem Public Key des Absenders entschlüsseln, selbst hashen, vergleichen
Zertifikat	Public Key + Angaben, signiert von einer CA → Authentizität
X.509	Standard für den Aufbau von Zertifikaten

Begriff	In einem Satz
Root Store	im System/Browser hinterlegte vertrauenswürdige Root-Zertifikate
PKI	RA prüft, CA stellt aus, VA bestätigt die Gültigkeit
CRL / OCSP	Sperrliste / Online-Abfrage gesperrter Zertifikate
QES	qualifizierte Signatur = rechtlich wie handschriftlich

✅ Lösungen

Ü1.1 Zwei von: **Einwegfunktion** (aus dem Hash lässt sich die Eingabe nicht berechnen), **Kollisionsresistenz** (zwei verschiedene Eingaben ergeben praktisch nie denselben Hash), **Lawineneffekt** (kleinste Änderung → völlig anderer Hash).

Ü1.2 1. Hashen ist **keine Verschlüsselung**: kein Schlüssel, nicht umkehrbar. 2. **MD5 ist veraltet und unsicher** (Kollisionen, extrem schnell zu berechnen). Besser: ein langsames Verfahren mit Salt, z. B. bcrypt, Argon2 oder yescrypt.

Ü1.3 Es gibt unendlich viele mögliche Eingaben, aber nur endlich viele Hashwerte (feste Länge). Irgendwann müssen sich also zwei Eingaben einen Hash teilen. Man macht die Zahl der Hashwerte nur so groß (z. B. 2^{256}), dass eine Kollision praktisch nicht zu finden ist.

Ü2.1 Ein Angreifer kann die Datei verändern, einen **neuen Hash** berechnen und beides weiterschicken. Der Empfänger merkt nichts. Der Hash muss deshalb mit dem Private Key des Absenders geschützt (signiert) werden.

Ü2.2 Anna signiert mit **ihrem eigenen Private Key**. Ben prüft mit **Annas Public Key**.

Ü2.3 Erfüllt: **Integrität** und **Authentizität** (🔎 plus Nichtabstreitbarkeit). **Nicht** erfüllt: **Vertraulichkeit**, denn das Dokument bleibt lesbar.

Ü3.1 Ein Zertifikat ist ein **Public Key mit Angaben zum Inhaber, signiert von einer vertrauenswürdigen Zertifizierungsstelle (CA)**. Es bestätigt, dass der Schlüssel wirklich zum Inhaber gehört.

Ü3.2 Vier von: Aussteller (CA), Inhaber, Public Key des Inhabers, Signaturalgorithmus / Hashfunktion, Gültigkeitsdauer, Seriennummer.

Ü3.3 Zwei von: Das Zertifikat ist **selbstsigniert** oder von einer **internen CA**, die auf dem Client nicht im Zertifikatsspeicher steht. Es ist **abgelaufen**. Der **Name passt nicht** zur Adresse (z. B. intranet statt

intranet.firma.de). Es wurde **gesperrt**. Die **Zwischen-CA fehlt** auf dem Server.

Ü4.1 RA (Registrierungsstelle) nimmt Anträge an und prüft die Identität. **CA** (Zertifizierungsstelle) stellt das Zertifikat aus bzw. signiert es. **VA** (Validierungsdienst) bestätigt auf Anfrage, ob ein Zertifikat gültig ist.

Ü4.2 Das Zertifikat muss bei der CA **gesperrt (widerrufen)** werden. Dann muss ein **neues Schlüsselpaar** mit neuem Zertifikat her. Browser erfahren es über die **Sperrliste (CRL)** bzw. **OCSP**, also über den Validierungsdienst.

Ü5.1 Die **qualifizierte elektronische Signatur (QES)**. Nötig sind ein **qualifiziertes Zertifikat** eines **anerkannten Vertrauensdienstes** (z. B. D-Trust) nach **Identitätsprüfung** und eine **sichere Signaturerstellungseinheit** (Chipkarte + Leser oder Fernsignatur beim Anbieter).

Ü5.2 Bei einem selbst erstellten Zertifikat hat **niemand die Identität geprüft**. Jeder könnte eins auf einen beliebigen Namen ausstellen. Die QES verlangt ein **qualifiziertes Zertifikat eines anerkannten Anbieters** und eine geprüfte SSEE.