x509-Zertifikate

Die Firma Netscape hatte einst das SSL-Verfahren (Secure Socket Layer) für sichere Webservices (https:// URL) erfunden und später als TLS (Transaction Layer Security) in die Public Domain gegeben. Die Basis dieses Verfahrens ist das Zertifikat nach dem **x509-Standard:**

Die x509- Norm beschreibt den Aufbau dieser Datei, die verpflichtenden und die möglichen Felder sowie die Syntaxregeln.

Die Grundidee ist dabei, ein digitales Dokument zu schaffen, das einem Personalausweis nachempfunden wurde:

1) Vertrauenswürdiger Herausgeber (Behörde beim Personalausweis)
2) Fälschungssicherheit (durch spezielle Druckverfahren, Papiersorte, Stempel)
3) Genormter Inhalt in maschinenleserlicher Form
4) Identifizierungsmöglichkeit (Lichtbild, Fingerabdruck, Beschreibung ...)

beim Zertifikat wird das umgesetzt durch

1) Herausgeber sind einige wenige internationale Firmen (Verisign, Thawte ...). Diese werden CA (Certificate Authority, Certification Authority, Certifying Authority) genannt. Leider sind 2011 mehrere solche CA wegen unzureichender Sicherheitsmaßnahmen gehackt und falsche Zertifikate erzeugt worden. Auf diese Art wollte der Iran (es gilt die Unschuldsvermutung) den verschlüsselten WWW-Verkehr zu Google abhören. Die Sicherheit der Zertifikate steht und fällt mit der Sicherheit der CAs, d.h. deren Signaturen und daran hängt die Sicherheit im Internet. Google und Netscape haben in der Folge angekündigt, einigen der CAs das Vertrauen zu entziehen (d.h. die Browser enthalten nicht mehr die öffentlichen Schlüssel dieser CAs).

2) Fälschungssicher: Die CA signiert das Zertifikat, der öffentliche Schlüssel aller vertrauenswürdigen CAs ist in jedem Browser hinterlegt, sodass alle Internet-User diese Signaturen verifizieren können.

3) Laut x509-Norm sind verpflichtend: Herausgeber (CN = Common Name), Seriennummer, Gültigkeitsdauer (von, bis), Inhaber (CN, Mail-Adresse ...), Beschreibung des Signaturverfahrens, Signatur der CA, …

4) Public Key des Inhabers. Damit lässt sich ein Challenge-Response-Verfahren durchführen

Challenge-Response-Verfahren: Man erstellt eine zufälligen String (Challenge) und übermittelt diesen an die Gegenstelle. Diese erstellt eine Signatur der Challenge und sendet diese zurück. Ist die Signatur korrekt (Überprüfung mit dem Public Key der Gegenstelle) so ist die Identität der Gegenstelle bewiesen (außer eine dritte Partei verfügt ebenfalls über den privaten Schlüssel!).

Ausführlicher Ablauf eines `https://web-mail.uibk.ac.at` Aufrufs:

Zuerst muss die IPv4 Adresse des Mailservers ermittelt werden. Dazu wird ein DNS-Server kontaktiert, der die IPv4 Adresse mitteilt. Darauf wird eine TCP-Verbindung zu dieser Adresse auf den https-Port 443 geöffnet (3-Way-Handshake des Osi-Layer 4). Gleich nach dem Aufbau dieser TCP-Verbindung wird der SSL/TLS Handshake durchgeführt:

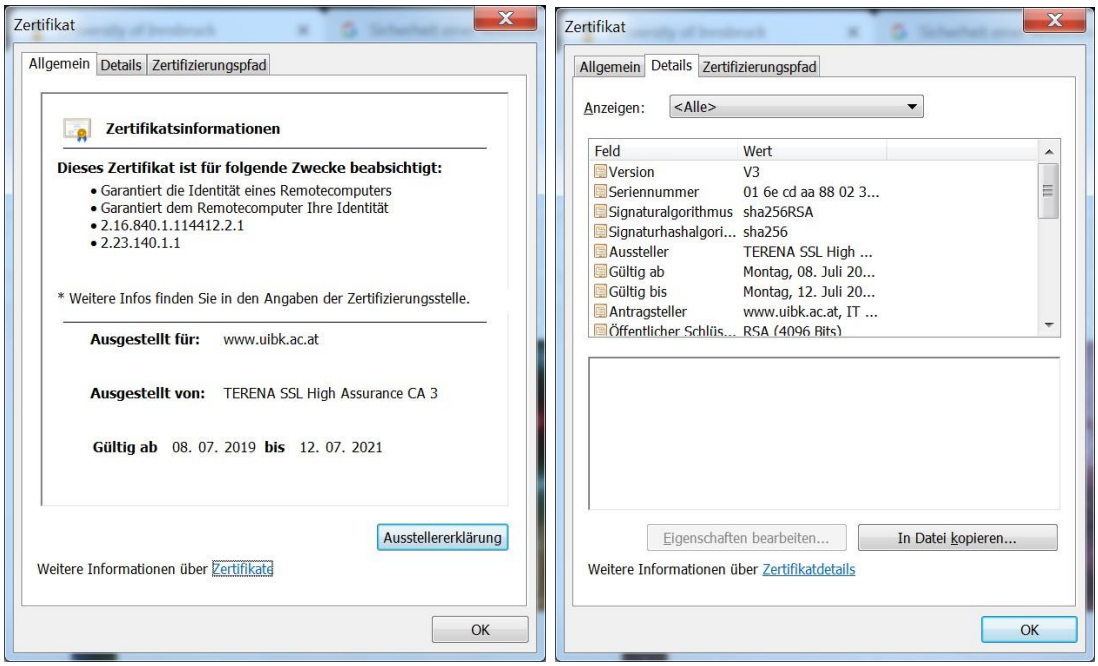
1) Client verlangt vom Server den Download seines x509-Zertifikats.
2) Server übermittelt das Zertifikat. Es erfolgt eine Überprüfung des Zertifikats:
   a. Ist der CA bekannt? (d.h. Public Key ist im Browser gespeichert)
   b. Ist die Gültigkeitsdauer passend
   c. Stimmt der CN des Inhabers mit dem DNS-Namen überein (CN == `web-mail.uibk.ac.at`); ein Seitenaufruf der Form `https://web-mail` führt meist zu einer Warnung, da das Zertifikat eben auf `web-mail.uibk.ac.at` lautet
   d. Ist die Signatur des Zertifikats ok
3) Zertifikat vom CA zurückgezogen?
4) Diffie-Helman (heute fast immer die EC Variante)-Handshake mit Authentisierung des Servers (durch Signatur)
5) Ableitung des endgültigen Session-Keys aus dem Diffie-Helman-Geheimnis.
6) Digitale Verschlüsselung des gesamten Netzwerk-Traffics (zumeist per AES).

Der Mozilla Firefox-Browser unterrichtet den Benutzer über die Sicherheitsalgorithmen, die beim Aufbau der gesicherten Verbindung eingesetzt wurden. Der Chrome Browser verhält sich hier mehr bedeckt und sagt nur sicher/nicht sicher ☹. Die nötigen Informationnen findet man aber etwas versteckt:

Chrome: TLS 1.2, ECDHE_RSA with P-256, and AES_256_GCM: Der Abruf der Seite erfolgte nach der Version 1.2 des TLS-Standards (1.3 ist die neueste und sicherste Variante), ECDHE_RSA steht für elliptic Curve Diffie-Helman mit RSA-Signatur-Validierung des Servers, P-256 ist die verwendete elliptische Kurve; Nach der Berechnung des Session-Keys erfolgt die beidseitige Verschlüsselung mittels AES (256 Bit-Schlüssel) im Galois-Counter-Mode.

Firefox: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256-Bit-Schlüssel, TLS 1.2: dieser String enthält dieselben Informationen, außer dass auch das verwendete Hashverfahren der Signatur mit angegeben wurde (SHA384).

In beiden Browsern kann man sich auch das Server-Zertifikat ansehen:



Die Windows Hilfe liefert folgende Infomationen zum Zertifikat:

Die Registerkarte **Details** enthält folgende Informationen zum Zertifikat:

- **Version**. Die X.509-Versionsnummer.
- **Seriennummer**. Die eindeutige Seriennummer, die dem Zertifikat von der ausstellenden Zertifizierungsstelle (Certification Authority, CA) zugewiesen wird. Allen von einer bestimmten Zertifizierungsstelle ausgestellten Zertifikaten wird eine eindeutige Seriennummer zugewiesen.
- **Signaturalgorithmus**. Der Hashalgorithmus, der von der Zertifizierungsstelle zum digitalen Signieren des Zertifikats verwendet wird.
- **Aussteller**. Informationen zur Zertifizierungsstelle, die das Zertifikat ausgestellt hat.
- **Gültig ab**. Zeigt das Anfangsdatum der Gültigkeitsdauer des Zertifikats an.
- **Gültig bis**. Zeigt das Enddatum der Gültigkeitsdauer des Zertifikats an.
- **Antragsteller**. Der Name der Einzelperson, des Computers, des Geräts oder der Zertifizierungsstelle, für die/den/das das Zertifikat ausgestellt wurde. Wenn sich die ausstellende Zertifizierungsstelle auf einem Domänenmitgliedsserver Ihres Unternehmens befindet, handelt es sich um einen definierten Namen, der innerhalb des Unternehmens verwendet wird. Andernfalls handelt es sich um den vollständigen Namen bzw. den E-Mail-Namen oder eine andere persönliche ID.

- **alternativer Antragstellername:** Alle DNS-Namen und IP-Adressen, für die das Zertifikat gültig sein soll. Die DNS-Namen dürfen dabei auch Wildcards enthalten wie z.B. `*.uibk.ac.at` (Gilt für alle Namen, die so enden!): Wildcard-Zertifikate. Enthält das Feld z.B. einen Aliasnamen des Servers oder dessen IP-Adresse nicht, so gilt es für diese Namen und IP-Adressen nicht und der Aufrufer bekommt eine entsprechende Warnung des Browsers.

- **Öffentlicher Schlüssel**. Der Typ und die Länge des öffentlichen Schlüssels, der dem Zertifikat zugeordnet ist.

- **Fingerabdruckalgorithmus**. Der Hashalgorithmus, der einen Digest (oder Fingerabdruck) der Daten digitaler Signaturen generiert.

- **Fingerabdruck**. Der Digest (oder Fingerabdruck) der Zertifizierungsdaten.

- **Anzeigename**. (Optional) Ein Anzeigename, der anstelle des Namens im Feld **Antragsteller** verwendet werden soll.

- **Erweiterte Schlüsselverwendung**. (Optional) Die Verwendungszwecke des betreffenden Zertifikats.

Ein Zertifikat kann zusätzliche X.509 v3-Erweiterungen enthalten. Diese werden, falls vorhanden, angezeigt.

Jeder Web-Browser enthält eine Liste von CA (d.h. deren Zertifikate), denen er immer vertraut. Der Browser vertraut auch Zertifikaten von Unter-CA, die sich mit einem Zertifikat der vertrauenswürdigen CAs auswisen können usw. Dem Zertifikat eines Webservers wird nur dann vertraut, wenn dieser eine lückenlose Reihe von vertrauenswürdigen CA und Unter-CA präsentieren kann (**Zertizierungspfad**), die seine Identität bestätigen.

Nehmen wir einmal an, ein Geheimdienst möchte alle Dateien kopieren, die Sie auf Google-Drive über eine https-Verbindung hochladen (die also sehr sicher verschlüsselt ist). Das beginnt schon mit dem DH-Handshake. Dieser ist in der Regel nicht zu knacken, außer der Angreifer gibt sich Ihnen gegenüber erfolgreich als Google aus, d.h. beweist durch ein gültiges Zertifikat, das auf Google ausgestellt ist, seine „Echtheit“. Der NSA fällt das nicht schwer, denn die könnte von Google die Herausgabe aller Schlüssel und Zertifikate verlangen (und haben das wahrscheinlich schon längst getan). Für den chinesischen (iranischen, indischen, …,österreichischen) Geheimdienst ist die Sache schon wesentlich schwieriger, weil Google denen (hoffentlich) nicht seine Geheimnisse verrät. Diese Angreifer brauchen ein auf Google ausgestelltes gültiges Zertifikat irgendeiner CA, denen mein Browser vertraut. Hier liegt auch die größte Schwachstelle der ganzen WWW-Sicherheit begraben: Jede CA kann beliebige Zertifikate ausstellen, denen alle Browser vertrauen. Und viele Geheimdienste haben solche willfährigen CAs zur Verfügung, hier unterscheiden sich die „guten“ und die „bösen“ Geheimdienste nicht wirklich.

In dem konkreten Fall mit dem iranischen Geheimdienst flog die ganze Geschichte auf, weil der abzuhörende Irani den Chrome Browser verwendete und dieser ganz genau weiß, welche

CA Google verwendet und bei der falschen CA sofort Alarm schlug. Hätte der arme Irani Firefox verwendet, wäre er wohl schon längst in irgendwelchen Kerkern verschwunden.

Als Gegenmaßnahme hat man eine Zeit lang das sogenannte Certificate-Pinning versucht, d.h. ein Webserver sagt, welche CA sein Zertifikat ausgestellt hat und der Client vertraut dann nur mehr einem Serverzertifikat von genau dieser CA und von keiner anderen. Allerdings entstanden dadurch mehr Probleme als gelöst wurden.

Chipcard, Smartcard, Signaturkarte, Handy-Signatur

Diese Karten dienen zur sicheren Aufbewahrung des private RSA-Keys. Im Normalfall darf der Private Key nie auszulesen sein. Jede Verwendung des Private Keys ist durch eine verpflichtende Eingabe eines PIN-Codes zu sichern. Bei Überschreiten einer Fehlschranke bei der PIN-Eingabe muss sich die Karte dauerhaft und unwiderruflich deaktivieren. All das und noch mehr sind die gesetzlichen Voraussetzungen für den Einsatz solcher Karten bei der qualifizierten Signatur, d.h. bei einer rechtsverbindlichen digitalen Unterschrift. Die Funktionen solcher Karten erlauben meist:

1) Auslesen des Zertifikats des Inhabers (ungeschützt)
2) Auslesen weiterer optionaler Zertifikate (ungeschützt)
3) Auslesen des Zertifikats der CA (ungeschützt)
4) Verschlüsselung eines Textes x mit dem Private Key (geschützt durch Pin-Eingabe). Hierdurch lässt sich z.B. eine Signatur erzeugen. Der Inhaber berechnet am PC eine Prüfsumme x (z.B. SHA256) über das zu signierende Dokument, sendet diese Prüfsumme x an die Smartcard, gibt den PIN ein und bekommt die Signatur aus der Karte zurück.
5) Entschlüsseln eines verschlüsselten Textes c mit dem Private Key (geschützt durch Pin-Eingabe). Hierdurch lässt sich z.B. der Sessionkey einer Nachricht entschlüsseln, mit dem man dann die Nachricht dechiffrieren kann. Heutzutage wird das nicht mehr benötigt, weil RSA nicht mehr zur sicheren Schlüsselübertragung eingesetzt wird und durch ECDHE ersetzt wurde.

Die Kommunikation mit der Smartcard erfolgt meist über einen Smartcard Reader. Selbstverständlich muss auch dieser Reader bestimmte Voraussetzungen erfüllen, um sicher zu sein. Sonst könnte ja z.B. die Pin-Eingabe aufgezeichnet werden etc. Auch das Umfeld (Web-Cams etc.) sollte überprüft werden. Bei der Einführung der qualifizierten Signatur sind leider längst nicht alle Schwierigkeiten erkannt worden und damit noch ungelöst: z.B. Ein persönliches Zertifikat (mit dem dazugehörigen Private Key) hat nur eine beschränkte Gültigkeit. Was geschieht mit erstellten Signaturen nach diesem Zeitraum? Sind diese dann noch gültig, wenn das signierende Zertifikat nicht mehr gültig ist? Was ist bei einem zurückgezogenen Zertifikat usw.?

Handy-Signatur:

Hierbei wird in Österreich die Speicherung der privaten Schlüssel an eine Firma (A-TRUST) ausgelagert und diese ist für die sichere Verwahrung verantwortlich. Der Client verifiziert sich gegenüber einer Seite (z.B. Finanz-Online, PVA, BVA, …) durch die Angabe der Handynummer und seines Signatur-Passwortes in einem Bildschirmformular der Fa. A-Trust (https://www.handy-signatur.at). Bei korrekter Eingabe sendet A-Trust einen Bestätigungscode per SMS an den User, mit dem dieser die Authentisierung bei Finanz-Online abschließen kann. Neuerdings steht dafür auch eine App zur Verfügung, mit deren Hilfe der Bestätigungscode vom Handy direkt an den Seitenbetreiber gesendet wird, nachdem das Handy einen Bildschirm-Punktcode gescannt hat oder der Handybesitzer der APp gegenüber seine Identität durch Fingerprint oder Face-Id nachgewiesen hat.

Auch in dieser Konfiguration gelangt der private Schlüssel niemals (hoffentlich) an Außenstehende, nicht einmal an den Inhaber. Das öffentliche Zertifikat dagegen ist sehr wohl über die Seiten der A-Trust zugänglich und wird auch in der App gespeichert.

Eigene Server-Zertifikate:

Will man selbst sichere Kommunikationsmöglichkeiten für einen eigenen Server einrichten, muss man sich ein Zertifikat für den Server besorgen oder ein solches selbst erzeugen. Sollen WWW-Browser diesem Zertifikat automatisch vertrauen, muss man sich an eine offizielle CA wenden. Es gibt heute durchaus Gratis-Zertifikate für solche Zwecke (z.B. von `letsencrypt.org`), allerdings besitzen diese oft nur eine kurze Gültigkeit (letsencrypt) oder werden von manchen Browsern nicht automatisch akzeptiert. Auf keinen Fall sollte dabei der private Schlüssel an die CA übermittelt werden oder von dieser generiert werden. Vielmehr muss man selbst ein RSA-Schlüsselpaar generieren und damit einen CA-Signing-Request erzeugen (dieser enthält nur den public Key und eine Signatur (zum Beweis, dass man auch den privaten Schlüssel besitzt), aber nicht den private Key selbst), für den die CA das Zertifikat generiert.

Aber selbst ohne vertrauenswürdige CA wird die Kommunikation perfekt abgesichert, d.h. die Vertrauensstellung der CA hat keinen Einfluss auf die Abhörsicherheit. Allerdings muss ein Client jedes Mal händisch den Zugriff auf Seiten mit solchen selbstgebastelten Serverzertifikaten erlauben.