Repetition

- Wie viel Inhalt können mit einem 1024-Bit Schlüssel übermittelt werden? 1024-Bit.
- Ist das Padding normiert? Ja, in einem RFC
- Grösse Integer? 32- / 64-Bit begrenzt durch Architektur, BigInteger :arrow_right: Spezialbehandlung
- Was ist ein qualifiziertes Zertifikat? Personenzertifikat

Padding

Ist der Text kürzer als der Schlüssel braucht es immer zwingend ein Padding. Ansonsten ist die Sicherheit der Nachricht gefährdet (Füllzeichen).

$x = a + bx + cy$
$a^x * a^y = a^{(x+y)}$
$a^{a_0} * a^{bx} * a^{cy} = a^{a_0+bx+cy} = a^x$
$a_0 = 0 \implies a^{a_0} = 0$
$(xy)^d \equiv x^d * y^d modn$
$C_1....C_2$
$C = \prod C(M_i)^{e^i} modn \rightarrow M = \prod M_i^{e_i} modn$

ASN.1 [Abstract Syntax Notation No. 1]

Wichtig: - Basic Encoding Rules (BER) - Distinguished Encoding Rules (DER)

Beschreibungsstruktur

Module

Modulname DEFINITIONS::=BEGIN EXPORTS export liste IMPORTS imports ....

www.oid-info.com

Sequence: 30h –> Tag: 0011 0000 Universal: 00 Zusammengesetzt: 1 Sequence: 16

Tag-Value: 30-0B-03-03-00-0F-C1-....

Sequence of Seuquence: 30 - L - 30 - ....

Objekt-Identifier

x x 40 + y 1x40 + 2 = 42

Zertifikate

Erzeugung

- Werden oft bei CA erstellt.

Ablauf:
1. Person geht zur RA für die initiale Registrierung (Intermidär).
2. RA bestätigt Identität und leitet den Request der CA weiter.
3. ....
x. PKCS#12-Container geht zurück an die Person (geschützt mit PIN)

Aufbau

ID + Signatur $sS_{p_{CA}}(ID)$ ObjectIdentifier :arrow_right: Signaturtyp + Algorithmus

Subject: Enduser oder Sub-CA

X.509 Zertifikate nach RFC 5280

Bestandteil X.500-Standards (Verzeichnisdienste), nicht immer optimale Lösungen, da bereits alt, Authentifizierungsstandard für Kommunikationsnetze, 3. Teil des Standards: Formate für digitale Zertifikate

Repetition

Hauptbestandteile Zertifikat: - tbs - AlgID - Sig ($ sS_a(tbs) $)

Erweiterungen / Extensions: Standard / Privat

Auflösung Zertifikatspfad: Via AuthorityKeyIdentifier

CA Zertifikat: Basic constraints, critical, Key Usage: critical (True, sonst immer false)

Dokumentunterschreibung: Key Usage: Content commitment

Attribut CRL: CRLDistributionPoint

Qualifiziertes zertifikat: Nur für natürliche Personen

Attributzertifikat

Erweiterung Zertifikat um weitere Attribute (z.B. auch zeitbegrenzt). Keine Identifikation, sondern Autorisierung

Analog zu SAML im Web-Bereich

Attribute müssen als OID hinterlegt sein

ASN1

objectDigestInfo:objectDigest: Hashwert Identifikationszertifikat

Sperrlisten (CRL)

crlDistributionPoint: URL für Sperrliste
Key-Attribut für Revokation: Seriennummer
crlEntryExtension: Bezogen auf jedes Zertifikat
crlExtensions: Extensions für CRL selbst CRL: Nicht notwendig, wenn normal abgelaufen CRL: Nur noch für kleinere CAs oder Sub-CAs, OCSP stattdessen

Verzeichnisdienst (OCSP)

RFC2560, via HTTP oder LDAP, Request-Signatur: optional

TBSRequest (n Requests) * Request * CertID * hashAlgorithm * issuernamehash * issuerkeyhash ("AuthorityKeyIdentifier", Hash Public key von Issuer) * serialnumber * singleRequestExtension*

OCSPResponse Immer signiert durch OCSP-Dienst

- SingleResponse
  - CertID
  - hashAlgorithm
  - issuernamehash
  - issuerkeyHash
  - serialnumber
  - certStatus
  - thisUpdate
  - singleExtensions*
- OCSPResponse (nicht signiert)
  - responseStatus (nicht optional, immer)

  - responseBytes (optional, Anfrage falsch, falsches Cert, Inhalt nur wenn Anfrage beantwortbar) :arrow_right: BasicOCSPResponse
- BasicOCSPResponse
  - tbsResponseData (signiert)

XCA

ZHAW-Root-CA - ZHAW-Sub-CA

Root-CA

1. XCA starte
    1. DB alege. -> Kes Passwort setze
    2. Certificates
    3. New Certificate
        1. Create a self signed certificate with the serial (self signed bedeutet es ist ein Root Zertifikat
        2. Signature algorithm: SHA 1
        3. Subject
        4. Internal name: ZHAW-Root-CA
        5. countryName: CH
        6. stateOrProvinceName: Zuerich
        7. localityName: Zuerich
        8. organizationName: ZHAW
        9. organizationUnitName: Certificate Services
        10. commonName: zhaw-root-ca
        11. Generate a new key
        12. Create
        13. Key usage
        14. Certificate Sign
        15. CRL Sign
        16. Key usage -> Critical
        17. Extensions
        18. Type -> Certification Authority
        19. Path length: 2
            - Leer = Unendlich lang
        20. Critical
        21. Subject Key Identifier
        22. Authority Key Identifier

23. Not after: 10 Jahre gültig
24. CRL distribution point -> Edit -> Add -> Content setzen auf: http://crl.zhaw-CA.ch (gibt es aber nicht) -> Apply
25. Finales OK

Sub-CA

- use this Certificate for signing: ZHAW-Root-CA
- Subject
  - Generate a new key
  - Ausfüllen
- Key usage
  - Critical
  - Certificate sign
  - CRL sign
- Extended key usage
  - Not critical
  - OCSP signing
- Extensions
  - Type Authority: Certification Authority
  - path length: 1, critical
  - subject key Identifier, authority key Identifier
  - Time range: kleiner als Root (5)
  - CRL distribution point (URI:http://crl.zhaw.ch/subca.crl)
  - OCSP (URI:http://ocsp.zhaw.ch/)
- Export DER

Enduser Zertifikat (EE-Cert)

- New Certificate
  - Use this Certificate for signing: Sub-CA
  - SHA1
  - HTTP_client
  - Generate a new key
  - Subject (ZHAW-Mail)
  - Key usage:
  - Digital Signature
  - Key Encipherment

- Data Encipherment
- Extended key usage:
- E-mail Protection
- Extensions
- End Entity
- Path length (leer)
- Subject Key Identifier
- Authority key identifier
- Time Range: 1 Year
- Subject alternative name: email:brundan1@students.zhaw.ch
- CRL distribution point (URI:http://crl.zhaw.ch/subca.crl)
- OCSP (URI:http://ocsp.zhaw.ch/)
- Export DER, PKCS12 (Private, Public, PW:DaniBrun), PKCS7 (with Chain, Public)

Timestamp-Dienst

Parameter: Hash Dokument, Ein Dokument aufs Mal mit Timestamp signieren, n können zurück kommen.

Hash

Kollisionsresistenz (Geburtstagsangriff), gleicher Hash für gleiche Eingabe,

SHA-1

Bläcke ä 512 Bit, Padding beim letzten Block (falls notwendig) bis zu 448 Bits, 64 Bit für Bit-Länge der ursprünglichen Nachricht, Grenze $ 2^64$

- Alle 512 Bit Blöcke in 16 32-Bit (Big Endian) unterteilen
- Epansion der 16 32-Bit Worte auf 80 32-Bit Worte
- Kompression über 4 Runden
- Addition der Hash-Werte :arrow_right: Hash-Wert (160 Bit)

#HMAC ##HMAC grössere Entropie , bessere Kollisionsresistenz

```
- Schlüssel-Länge (aufgepaddet mit 0)),
- XOR mit ipad (Konstant)
- + Nachricht
- Hash-Wert
- + Schlüssel XOR opad (Konstant)
```

SSL

Je höher die Schicht, desto unsicherer

Übung

Zertifizierungsanfrage (Public Key, Subject Information)

- Neues Zertifikat zu abgelaufenem Zertifikat, Unterschreiben mit altem Private Key

```
openssl
req -nodes -new -newkey rsa:2048 -outform DER -out csr.der
./dumpasn1 al csr.der
```

- -nodes: optional, Private Key wird auch in die csr.der geschrieben ##Protokollstack Ziel / Fixpunkt: DNS-Name (localhost:1234/test), Schichten: 5, 6, 7, Daten über Schicht 5 als SSL-Record, SSL beinhaltet / verwendet nirgends IP-Adressen, nur DNS

SSL/TLS Record Header: Handshake noch unverschlüsselt **Handshake:** Client bleibt anonym (Browser kein Zertifikat), Diffie-Hellmann: Nur wenn Client anonym, sonst nicht, sonst: immer gleicher Key, heute wird Ephemeral Diffie-Hellmann verwendet (DHE_DSS, DHE_RSA), EDH: Vorteil: immer neuer Session Key (auch wenn client gehackt), RSA: Session Key voll von Client abhängig

Protokoll

- hello_request (Aushandlung neuer Session, auch während laufender Kommunikation), Server zu Browser
- client_hello (Beginn Aufbau SSL-Session)
- Erweitertes client_hello
- server_hello
- Erweitertes server_hello
- certificate (Zertifikat inkl. Chain)
- server_key_exchange (bei RSA braucht es dies nicht, nur bei Diffie Hellmann und Diffie Hellmann Ephemeral)
- certificate_request
- server_hello_done
- certificate_verify (Nur client, Bildung Signatur Client, keine Verifikation Zertifikat auf Client)
- client_key_exchange
- finished

Alert Protokoll

Eigener Payload, kann auch während Chiffrierung kommen, innerhalb eines SSL-/ TLS-Records, Level: Warning, Fatal

Application Data Protokoll

Record-Layer 5, Transport Anwendungsdaten ohne Betrachtung Inhalt

Kryptografische Komponenten von SSL und TLS

Schlüsselerzeugung

| :arrow_right: Konkatenation

IPsec

Je weiter oben im OSI-Modell, desto unsicherer

Standardisierung

IKE

Handshake / Initialisierung

IPsec Protokolle

Kein Client / Server, beide gleichberechtigt, Multi-VPN möglich

IPv6

Header: unter Umständen: verkettete Liste Mask: Eingeführt wegen zu wenigen IP-Adressen, eigentlich nicht mehr notwendig

Übertragungsmodi

Tunnel-Modus: 2 IP-Adresse: 1 Klartext, 1 Verschlüsselt, bei beiden ESP möglich
Transport-Modus: Nur 1 IP-Adresse

Teilprotokolle

Authentication Header

Authentifiziert (z.B. MAC), nicht asymmetrisch (zu langsam), Tunnel- und Transportmodus, nicht verschlüsselt

SPI: Security Parameter Index, analogie zu Cyphersuite, Aushandlung, Zuordnung Security Assoziation

Encapsulating Security Payload

Verschlüsselt, Transport- und Tunnelmodues

Einbindung ESP in IPv4 (Transportmode) next: Verweis auf TCP-header innerhalb des Payloads

Einbindung ESP in IPv4 (Tunnelmode) next: Verweis auf IP (innere IP) im Payload

IPsec Management

Security Association Database und Security Policy Database pro VPN-Receiver

ISAKMP

Version egall, universell

IKEv1

Authentisierung

- Variante 1: Via Signatur (explizit)
  - Nachricht 1
  - Header: IKE-Header (ISAKMP, S. 27), Next Payload
  - SA: Security Assoziation (als Payload, S. 30)
  - Proposal 1: Proposal Payload (Einleitung Parameter, S. 30, SPI enthalten)
  - Transform 1: Transform Payload (Parameter, S. 31)
  - Antwort 1
  - Header: ISAKMP-Header

  - SA: Auswahl SA
  - Proposal: Ausgewähltes Proposal
  - Transform: Ausgewählter Transform
  - Nachricht 2
  - Header: “”
  - KEi: Diffie-Hellmann-Parameter (3)
  - Nonce: Zufallszahl
  - Certificate Request: Optional
  - Antwort 2
  - Header: “”
  - KEr: Diffie-Hellmann-Parameter
  - Nonce
  - Certificate Request: optional
  - Nachricht 3
  - Header
  - Payload: Verschlüsselt (Diffie-Hellmann ist ausgetauscht)
    * IDi
    * Certificate: Optional
    * Signatur i (Authentifizierung)
  - Antwort 3 -Header -Payload: Verschlüsselt
    * IDr
    * Certificate: Optional
    * Signatur r: (Authentifizierung)
- Variante 2: Authentisierung mit Public Key Verschlüsselung (implizit)
  - Nachricht 1 und Antwort 1 analog
  - Nachricht 2
  - Header
  - KEi: Diffie-Hellmann-Parameter (3) für Session-Key
  - Hash(Certifikat): optional, Hash von Zertifikat
  - IDi: Verschlüsselt mit Public-Key von Responder
  - Ni: Verschlüsselt mit Public-Key von Responder
  - Antwort 2
  - Header
  - KEr: Diffie-Hellmann-Parameter für Session-Key
  - IDr: Verschlüsselt mit Public-Key von Initiator
  - Nr: Verschlüsselt mit Public-Key von Initiator
  - Nachricht 3
  - Header
  - Hash_i: Verschlüsselt mit Diffie-Hellmann / Session-Key
  - Hash_r: Verschlüsselt mit Diffie-Hellmann / Session-Key
- Veriante 3: Pre-Shared Key

- Nachricht / Antowrt 1 und 2: analog
- Nachricht 3:
  - Header
  - KEi: Diffie-Hellmann (3)
  - Ni
- Antwort 3:
  - Header
  - KEr: Diffie-Hellmann (3)
  - Nr
- Nachricht 4:
  - Header
  - Hash: Identifikation (mit Shared-Secret)
- Antwort 4:
  - Header
  - hash: Identifikation (mit Shared-Secret)

Aggressive-Mode

Schneller (nur 3 Nachrichten für Etablierung) und unsicherer, "sinnvoll" bei variablen / dynamischen IP-Adressen - Nachricht 1 und Nachricht 2 verkettet & zusammengefasst - Antwort 1: Antwort 1, 2 & 3 verketten & zusammengefasst - Nachricht 2

Gleiche Modes (Variante 1, 2 und 3 für Aggressive Mode), Variante 3 (Pre-Shared) im Aggressive-mode nicht verwenden! -> Knackbar

IKE Quick Mode

IPsec-Schlüsselaustausch, anschliessend alles verschlüsselt (Key-Exchange bereits stattgefunden), braucht keine Authentisierung mehr, Wesentlichster Punkt: Austausch neues Key-Material, mit PFS oder ohne

NAT-Traversal

NAT/PAT und IPsec

NAT: Adress-Übersetzung (Kein Subnetz!), PAT: Port-Übersetzung, NAPT: Beides Zusammen

- NAT: Tunnelmode funktioniert, Transportmode funktioniert nicht (AH: Problem MAC, nur ein IP-Header, wenn transferiert -> MAC stimmt nicht mehr)

UDP-Einkapselung

Kapselung von IKE- und IPSec-ESP-Paketen: ISAKMP und Payload müssen gekapselt werden, Unterscheidung, nur ESP mit Transport- und Tunnelmode, AH läuft nicht

- NAT-T: UDP Encapsulated ESP Header format
- NAT-ESP Marker: IKE Header Format für Port 4500
- NAT-T: ESP Transportmodus
- NAT-Traversal: EsP Tunnelmodus
- NAT-Keepalive Paket

NAT-T

Vendor-ID: Hashwert von "RFC 3947" aka "Ich kann NAT-T", NAT-Discovery Payload, Port: Meist 500 UDP am Anfang,

IKE-Main-Mode: - Nachricht 1: - Header: ISKAMP - SA: Security Assoziation - Vendor-ID (Initiator -> Responder, ich kann NAT-Traversal) - Antwort 1: - Angabe eines neuen Ports für Zukunft - Header, SA, Vendor-ID - Nachricht 2: - Header, Diffie-Hellmann, Ni - NAT-D(r): IP / Port von Gegenseite
Stimmt überein: Kein NAT, sonst Keepalive-Pakets senden damit der Router den Eintrag in der Tabaelle behält. - NAT-D(i): Eigene IP / Port - Antwort 2: - Header, Diffie-Hellmann, Nr - NAT-D(r), NAT-D(l) -> Check off NAT-Fähig) - Nachricht 3: - Non-ESP-M, Header (ISAKMP), ID(i), Cert, Sig(i) - Antwort 3: - Non-ESP-M, Header, ID(r), Cert, Sig(r)

NAT-T: IKE Aggressive Mode: Analog

NAT-T: IKE Quickmode für ESP Transport-Mode NAT-OA-Payload: Original IP-Adresse, Quickmode bevor ESP Transport-Mode

Internet Key Exchange Protocol Version 2 (IKEv2)

Init, Authentifizierung, Mehrere Kanäle auf gleicher Authentifizierung (Childs, ähnlich Quick-Mode), Informationsautschausch separat

IKE_SA_INIT

Cookies gegen DOS-Attacken
Phase 1 - Nachricht 1: (Keine Authentifizierung) - ISAKMP-Header - Security Assoziation - Diffie-Hellmann - Nonce - Antwort 1: - ISAKMP-Header - Security Assoziation - Diffie-Hellmann - Nonce - Cert (optional)

IKE_AUTH

Phase 2, Alles verschlüsselt, aber noch nicht authentifiziert - Nachricht 1 - ISAKMP-Header - IDi - Cert: optional - Cert-Req: optional - IDr: optional - AUTH: Authentisierungs-Information - SAi2: Ausgehandelte SA, Bestätigung - TSi: Traffic-Selektoren - TSr: Traffic-Selektoren - Antwort 1 - ISAKMP-Header - IDr - Cert: optional - AUTH - SAr2 - TSi: Anpassbar vom Responder - TSr: Anpassbar von Responder

CREATE_CHILD_SA

Nochmals Diffie-Hellmann, Sitzungsschlüssel, Pro Child: Nachricht / Schritt 3

Recap IPSec

IKEv1 / IKEv2: Authentifizierung, Schlüsselaustausch, implizit (über Entschlüsselung), explizit (via Signatur), Diffie-Hellmann, Main- / Aggressive / Quick-Mode, Aggressive: Zusammenfassung Schritte, Shared-Secret, Pre-Shared-Key + Aggressive: Knackbar (Wörterbuch, Known-Cypher, Orakel, Bruteforce), Agressive-Mode (Bei dyn IP), Adressierung via SIP, ISAKMP-Header: UDP Port 500, Verhinderung Replay: Via Cookies, Header + Payload als Verkettete Liste, Quickmode IKEv1 - Create Child SA in IKEv2, IKEv2: INIT (SA festlegen, Schlüsselaustausch), IKE_AUTH, NAT-Traversal: AHA geht nicht, Modes: Transport / Tunnel (funktionieren, bei Transport: IP nachreichen), NAT: Kapselung IP / Port in UDP, IKEv2 Traffic-Selektoren: ähnlich Firewall / Port-Filter-Firewall (Beidseitig), IKEv1: Security Policy (nur beim Absender)

Kerberos

Basis: Symmetrische Verschlüsselungsverfahren

Key Distribution Center (KDC): Integriert AD, Authentication Server (AS), Ticket Granting Server (TGS), gemeinsame Datenbank

Ablauf: 0. Voraussetzung: Pre-Shared-Secret oder Public-Key bei Schritt 1. 1. Client-Request auf KDC-AS - ID-Client, ID-TGS 2. Response von KDC-AS - $A_{TGS,C}|T_{TGS}$ - $A_{TGS,C} = eK_{C_KDC}(K_{C,TGS})$ (Kann vom Client geöffnet werden) - $T_{TGS} = e_{TGS_KDC}(K_{C,T||TimesGS}||I||ID_C)$ 3. Würfeln 4. Client-Request an KDC-TGS - ID_{AP} (ID Applikationsserver, Principalname, Realm) - $B1_{C,TGS} = eK_{C,TGS}(ID_C||ZT)(in2.ErhaltenerSchlüssel)$ - $T_{TGS} = eK_{TGS,KDC}(K_{C,TGS}||I||ID_C||Times)$ (aus Schritt 2, Weiterschicken, Wenn auspacken möglich -> Authentifiziert) 5. Response von KDC-TGS - $B2_{C,TGS} = eK_{C,TGS}(K_C, AP)$ (Schlüssel aus Schritt 2, 4) - $T_{AP} = eK_{TGS,AP}(K_{C,AP}||I||ID||Times)$ 6. Client-Request auf

Application Server - $C = eK_{C,AP}(ID_C||ZT||Subkey||Seq)$ - $T_{AP} = eK_{TGS,AP}(K_{C,AP}||I||ID||Times)$ 7. Response vom Application Server - $AP_REP = eK_{C,AP}(ZT,Subkey,Seq)

Pre-Shared-Secret zwischen TGS und Application-Server

Realm: "Domäne", Domänenüberlagernd / -übergreifend, mögliche Hierarchische Verschachtelung, Name: immer mit Grossbuchstaben
Kerberos Principals: ID + Geheimnis, Verwaltung via kadmin, innerhalb REALM
User Principals: ID aus user@REALM, Geheimnis: Passwort, Instance / rolle: user/instance@REALM (Zuweisung von Rollen)
Service Principals: ID: Serviceangabe/FQDN des Servers, service/dns.domain.name@REALM (service: http, ftp, pop), Geheimnis: Zufallswert, auf Server hinterlegt (keytab)
KDC: AS, TGS, Besondere Sicherheitsmassnahmen, DB mit User- / Serverprincipals, Redundanz via Weitere KDC-Server (Slaves), Abgleich mit Master (Vorzugsweise via IPSec)
Voraussetzungen für Kerberos: Zeitsynchronisation (max. 5 Min Differenz), Korrekte Einträge im DNS, Spezielle Absicherung, Mind. 2 KDC (Master, Slave), nur Kerberos V, Backup für REALM-DB

Schwächen: Passwort erraten, Synchronisation, Authentisierung nur einseitig

Prüfungsbesprechung

Aufgabe 1

- Stufe D
- CIA
- Signatur erstellen, Verschlüsseln, Session-Key transferieren
- Digital Signature, Key-Encipherment, Content-Commitment
- id-kp-emailProtection

Aufgabe 2

- Unterschied Anwendung Privater Schlüssel explizit und implizite Authentifizierung Explizit: Private Key zum verschlüsseln, Implizit: Public Key zum verschlüsseln
- Anforderungen an ausgetauschte Authentifizierungstoken
  Typischer Angriff: Replay, Random-Number
- Verfahren / Funktion wird digitale Signatur aufgebaut (nach ISO/IEC 9798 Notation)
  SSA(n)) = eSa(H(n))

- Was ist eine Authentifizierung:
  Prozess in dem Sicherheit über Identitäts-Behauptung gewonnen wird
- Identifikator: AHV-Nummer

Aufgabe 3

- Grundobjekte AD-Baum
  Knoten, Blatt
- oberstes Element in Verzeichnisbaum?
  Root
- Distinguished Name "Nicole Roux":
  dn={cn=Nicole Roux,ou=Buchhaltung,c=Example,c=ch}
- Relativ DSN "Frank" unter Example:
  rn={cn=Frank Roda,ou=Management} (Kontext: Eample)
- Kontext

Aufgabe 4

- Anforderungen an fortgeschrittene Signaturen
  - Siehe Script Kurs3_v2
- Zusätzliche Anfroderungen bei Qualifizierte Signatur
  Qualifiziertes Zertifikat (Natürliche Person), erzeugt mit sicheren Signaturerstellungseinheit (SSCD)
- Schweizweit geregelte elektronische Signatur Qualifizierte Signaturen
- Qualifiziertes Zertifikat für Web-Server? nein
- Bitmap Unterschrift als elek. Siggnatur? Ja
- RFC-Standard für Erweiterung, wie wird Segemntqualifier
  Ja, jeder kann qualifiziertes Zertifikat ausstellen

Aufgabe 5

- Typ PSE im Firefox / Microsoft?
  Software
- Standard API Smaartcard Browser
  PKCS#11
- PKCS#7

- Backup, rollen

Aufgabe 6

- Unterschied ASN.1 SEQUENCE, SET
  Reihenfolge
- Beschreibung Element “Description”
- Klassen Bezeichnerfeld, strukturierte Klasse
  6, Constructed, Private, Application
- Bit-String DER-Kodierung für ’10011’B

Aufgabe 7

- Zertifikatshierarchie grafisch
  Root, CA, Cert
- Erweiterung für Darstellung Zertifikatspfad im Browser , Eintrag in Hierarchie
  Subject-Key-Identifier, Authority-Key-Identifier
- Was steht im Subject- / Issuer-Attribut der SwissMarathon Root CA?

- 2 x das gleiche
- Unterschieden sich die öffentlichen Schlüssel in einem Root-Zertifikat Nein
- Was steht im Subject- / Issuer-Attribut Ihres Client-Zertifikats?
  Subject: E-Mail, Land, Name (DN)
- Erweiterung mit Attributfeldern in Zertifkaten wird immer benötigt? Basic Constraints, Wichtig, CA + Pfad-Länge
- Attribute für Gültigkeitsdauer von 3 Jahren in Ihrem Zertifikat, ausgehend von Heute

- Erweiterung für URL für OCSP-Dienst

- Erweiterung URL für Verzeichnisdienst CA?
  Subject-Info-Access

Aufgabe 8

SSL/TLS-Authentifizierung, Endbenutzerzertifikat, Portalzugang mit Authentifizierungssignatur mit SHA1, SHA256, 2 Verschiedene Hashverfahren - Mit Endbenutzerzertifikat möglich?
Ja - Wenn ja, wo Hashfunktion spezifiziert?

Aufgabe 9

- Welches Feld CRL Seriennummer und Datum wiederrufenes Zertifikat
  Revoke Certificate

- Geläufige optionale Erweiterung?
  Reason

Aufgabe 10

- Mind. Inhalt Antwort in OCSP-Anfrage? Status
- Welches Attributfeld Referenzwert Wiederruf Zertifikat, wie heisst er?
  SingleResponse: Serial-Nummer, Status
- OCSP-Anfrage mehrere oder ein Zertifikat bezüglich Wiederruf anfragen?
  Nur eines
- Feld Wiederrufsgrund OCSP-Anfrage?
  RevokedInfo
- OCSP-Antwort mit entsprechenden Attributen für ihr Client-Zertifikat

Chip-Karten

ID-1: z.B. Krankenkassen-Karte, ID-00: z.B. SIM-Karte, ID-000: Nano, Micro SIM-karten

Kartentypen: Reine Speicherkarte (256 Byte EEPROM), Intelligente Speicherkarte (Sicherheitslogik R/W –> SIM), Mikroprozessorkarte (Smartcard, Speicher + CPU mit OS), Kryptokarte (Prozessorkarte + Coprozessor für Krypto), Multi-Applikations-Karte (z.B. JAvaCard mit JVM)

Karte: Einschub in Lesegerät, immer Reset

Kontaktlose Chipkarten: Induktive Kopplung (über spule, z.B. via Amplitudenmodulation), Kapazitive Koppelung (leitende Fläche, Kondensatorplatten, nur für Datenübermittlung), Kombination möglich

Aufbau einer Chipkarten-Applikation: Sequentiell APDU-Befehle absetzen, nacheinander, State-Machine aus Sicherheitsgründen, Hash-Wert der PIN auf Karte, Klass 2/3-Leser (2: Nur Tastatur, 3: + Display), Zertifiziert, sodass Daten nicht ausgelesen werden können, APDUs auf Schicht 7, nur Schichten 1,2,7, 9600 Bit/Sekunde, nicht für grosse Datenmengen,

Kommunikation mit dem Terminal

- Request:
  - CLA: Normal 00
  - INS: Instruktion für Chip (1 Byte)
  - P1, Pa: Parameter für Befehl / Instruktion
  - Le: Längenfeld der mitgesendeten Daten

  - Data: Daten (max 64 kBytes)
  - Le: Länge erwarteten Daten
- Response
  - Data: Datenfeld (otpional)
  - SW1, SW2: Returncodes ###Protokoll T0 Kleinste übertragene Einheit: Byte, Bitorientiert, Rückwärtskompatibilität, max 256 Byte übertragen, Layer 2

Protokoll T1

Kleinste übertragene Einheit: Datenblock, Abbildung APDU, sicherer, 64 kByte transferieren

Chipkarten OS

Native (C) und interpreterbasierte OS (C, Java), JAvaCard, BasicCard, Multos

Struktur des Dateisystemes

Aufbau: Master-File (3F00), Dedicated Files (Ordner), Elementary Files (Dateien), EF unter MF oder DF, **File Identifier:** Adressierung Files (meistens 2 Byte oder 1 Byte), zum Tei Identifier fix (z.B. MF: 3F00), **Application Identifier (AD):** DF einer AID zugeordnet, Java-Card: Security Domains, Vergabe via AID

Dateistrukturen EFs

Transparent: Read / Write / Update Binary, ähnlich Floppy-Disk, keine innere Struktur, byte- oder blockweise **Linear Fixed:** Record-Struktur, feste Struktur mit fester Länge, Struktur im FCP **Linear variable:** Record-Struktur, variable Länge Struktur, Nicht Änderbar (Löschbar) nach dem Anlegen **Cyclic:** Record-Struktur, fix, zyklisch organisiert, Ring-Puffer (überschreiben), z.B. für Logs / Protokoll

Aufbau Dateiverwaltungsinformation

File Control Parameter (FCP): Parametrisierung DF, Datenstruktur, für alle File-Typen, Starre Struktur / Record, Internal-Files (nicht zugreif- / lesbar, z.B. Krypto, OS), Working EFs: Nach aussen sichtbar, les- / schreibbar

Sicherheitsstatus

Get-Challenge: Zufallswert generieren, internal / external Authenticate, A challenge B via T,

Sicherheitsmechanismen

Entity-Authentifizierung mit Passwort / Schlüssel, Daten-Authentifizierung, Daten-Entschlüsselung

APDU

Zugriff auf Dateien über Dateibezeichner, Pfad, SFID, DF-name

Java-Card

Verschiedene Security-Domain, Issuer-Security-Domain, Weitere Domain, jede SD: AID, Applets können in SD geladen werden, SD sind getrennt und isoliert, Laden der Apps auf Karte: Global Plattform Standard, Pro SD: Authentifizierunsschlüssel

Prüfungsbesprechung Prüfung 2

Aufgabe 1

- Wird Doument zuerst signiert und nachher Zeigestempelt?
  Zuerst Zeitstempel, dann Signatur
- Können Sie in einer Anfrage mehrere Dokumente zur Zeitstempelung einbeziehen.
  Nein
- Beschreiben Sie kurz die Erstellung eines Zeitstempels. Hash der Nachricht, Zeitstempel erzeugen, Signierung Hash + Zeitstempel, Anhängen Zeitstempel

- Zälen Sie kurz die Attributfelder einer Zeitstempelanfrage auf? Bennenen Sie Attributfeld für zeitzustempelndes Dokument
  version, MessageImprint, hashAlogrithm, hashedMessage, certReq
- Schutz for Replay
  Nonce

Aufgabe 2

- Welches Trustmodell wird insbesondere durch PGP unterstützt?
  Benutzer Trustmodell
- Zählen Sie diem öglichen Gültigkeitsmodelle für CA-Zertifikate auf.
  Schalen, Hybrid, Kettenmodell
- Bei welchem Gültigkeitsmodell endet die Gültigkeitsdauer eines Endbenutzerzertifikates spätestens mit der Gültigkeitsdauer der ausstellenden CA? Schalenmodell

Aufgabe 3

- Wie heisst das Protokoll auf dem eine SSL/TLS Nachrichtenübertragung aufgebaut wird?
  Record-Protokoll
- Zählen Sie die Schritte auf, wmit welchen die Anwendungsdaten in das oben bezeichnete Protokoll eingebettet werden. Fragmentierung, Komprimierung, MAC+Verschlüsselung
- Wie heissen die nächsthöheren protokolle bezogen auf Aufgabe 4a) und in welchem Feld werden diese Punkte definiert? Hand-Shake, Content-Type
- Benennen Sie die Nachrichtentypen, welche sich beim SSL/TLS-Kommunikationsaufbau zwischen den Verfahren “Client anonym”, “Client Auth mit Zertifikat” unterscheiden. Anonym ohne: CertificateRequest, Certificate, CertificateVerify
- Bescheiben Sie kurz chronologisch, was beim handshake im Teilschritt “certificate_verify” für Verfahren durchgeführt werden müssen, wenn DHE angewendet werden soll. Signatur verifizieren von sämtlichen Hand-Shakes von allen bisher stattgefundenen Kommunikationen, Input für Verifikation: Hashes von allen Handshakes (von Client, Prüfung gegen auf Server hinterlegten)