

# IT-Security Tutorübung 09

---

Dorian Zedler

18. Dezember 2023

Technische Universität München

Aufgabe 1

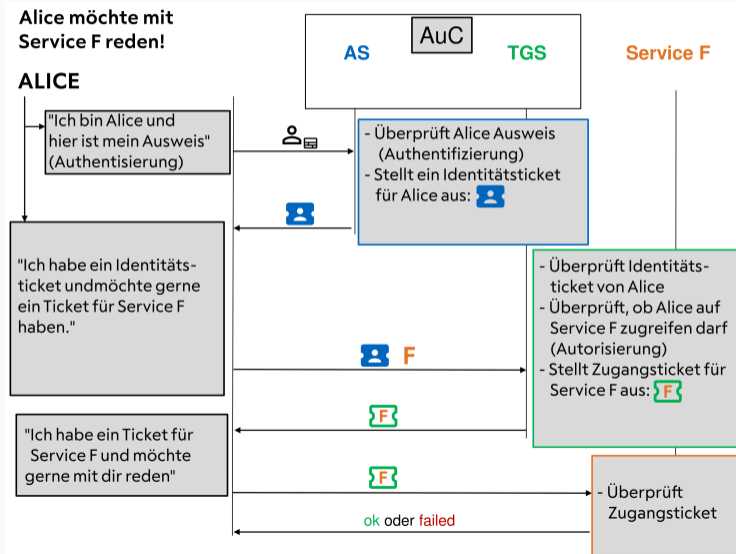
Aufgabe 2

- Kerberos
- PKI

# Aufgabe 1

---

# Kerberos - Vereinfacht



## Aufgabe 1a - Kerberos - Was soll das?

- a) Welche Aufgaben erfüllt das Kerberos-Protokoll?
- Authentifizierung
  - Autorisierung
  - Verschlüsselung / Schlüsselverteilung
  - Integritätsschutz
  - Alles **ohne** asymmetrische Kryptografie!

## Aufgabe 1b - Kerberos - Ticket

b) Beschreiben Sie die Funktion eines *Tickets* im Kontext von Kerberos!  
Woher stammt es und aus welchen Bestandteilen besteht es?

- Wird vom Authentifizierungsservice (AS) oder Ticket granting service (TGS) ausgestellt
- Berechtigt zur Nutzung eines Dienstes
- Transportiert den gemeinsamen Sitzungsschlüssel
- Ticket =  
(Service<sub>ID</sub>, Principal<sub>ID</sub>, Principal<sub>(IP-)</sub>Adresse, Timestamp, Lifetime,  $K_{PS}$ )

# Kerberos - Komplett

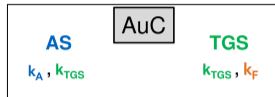
Alice möchte mit Service F reden!

**ALICE**  $k_A$

$k_A = \text{KDF}(\text{password}, ..)$   
 $c = \text{AES}_{k_A}(\text{current\_time})$   
 Generiert Nonce N1

$k_A = \text{KDF}(\text{password}, ..)$   
 $k_{A,TGS}, N1, TGS = \text{AES}_{k_A}^{-1}(c1)$   
 Prüft N1, TGS  
 Generiert: Nonce N2  
 Erstellt:  $\text{Authent1}^A = (A, ip, ts)$   
 $c3 = \text{AES}_{k_{A,TGS}}(\text{Authent1}^A)$

$k_{A,F}, N2, F = \text{AES}_{k_{A,TGS}}^{-1}(c4)$   
 Prüft N2, F  
 Erstellt:  $\text{Authent2}^A$   
 $c6 = \text{AES}_{k_{A,F}}(\text{Authent2}^A)$



Prüft  $\text{current\_time} = \text{AES}_{k_A}^{-1}(c)$   
 Generiert:  $k_{A,TGS}$   
 $c1 = \text{AES}_{k_A}(k_{A,TGS}, N1, TGS)$   
 Erstellt Ticket  
 $\text{Ⓜ} = \text{AES}_{k_{TGS}}(TGS, A, ip, ts, k_{A,TGS})$

$(.., k_{A,TGS}) = \text{AES}_{k_{TGS}}^{-1}(\text{Ⓜ})$   
 $\text{Authent1}^A = \text{AES}_{k_{A,TGS}}^{-1}(c3)$   
 Prüft  $\text{Authent1}^A$   
 Generiert:  $k_{A,F}$   
 $c4 = \text{AES}_{k_{A,TGS}}(k_{A,F}, N2, F)$   
 Erstellt Ticket  
 $\text{Ⓜ} = \text{AES}_{k_F}(F, A, ip, ts, k_{A,F})$

$(.., k_{A,F}) = \text{AES}_{k_F}^{-1}(\text{Ⓜ})$   
 $\text{Authent2}^A = \text{AES}_{k_{A,F}}^{-1}(c6)$   
 Prüft  $\text{Authent2}^A$

$c, A, TGS, N1$

$c1, \text{Ⓜ}$

$\text{Ⓜ}, N2, c3, F$

$c4, \text{Ⓜ}$

$\text{Ⓜ}, c6$

ok oder failed

**Service F**  
 $k_F$



- c) Wozu dient der *Authenticator* (z.B. *Authent1*) im Kontext von Kerberos?
- Nachweis, dass Principal (z.B. Alice) berechtigt ist, das Ticket zu nutzen
  - Beweist, dass Principal den gemeinsamen Sitzungsschlüssel (z.B.  $k_{A,F}$ ) kennt
  - Schützt durch enthaltenen Timestamp vor Replay-Angriffen

- d) Nennen und erklären Sie **zwei** Probleme, die beim Einsatz eines KDC entstehen!
- 1) **Kein Perfect Forward Secrecy:** Kompromittierung des Schlüssels eines Prinzipals (z.B.  $k_A$ ), kann alle seine Kommunikation entschlüsselt werden
  - 2) **Single Point of Failure:** KDC kennt alle Schlüssel und muss jede Kommunikation initiieren

- e) Könnten AS und TGS in einen Server zusammengefasst werden?
- **JA**, sie könnten beide auf demselben Server laufen
  - Erfüllen aber unterschiedliche Aufgaben und werden deshalb im Protokoll getrennt

- f) Wozu wird Alice im ersten Schritt mittels der Nachricht  $c$  authentifiziert?
- Alice authentifiziert sich gegenüber dem AS und beweist, dass sie das Passwort kennt
  - Ohne PreAuth könnte ein Angreifer ein  $C_1$  für Alice anfordern und das Passwort lokal bruteforcen
  - Mit PreAuth müsste der Angreifer zunächst eine Man-in-the-Middle-Position erreichen
  - PreAuth ist in Kerberos trotzdem optional

- g) Wie wird die Integrität der Nachrichten in Kerberos sichergestellt?
- Verschlüsselte Nachrichten haben eine HMAC Checksumme über den Klartext (MAC-then-encrypt)
  - Früher auch Checksummen ohne Schlüssel (z.B. CRC32) erlaubt, aber inzwischen deprecated

- h) Die an Kerberos angeschlossenen Services müssen einen Replay Cache führen. Wozu ist das notwendig?
- Szenario:
    - Alice sendet Authent (c6) und Ticket an den Service
    - Alice sendet einen Befehl für eine Zahlung an den Service
    - Diese beiden Nachrichten werden von einem Angreifer aufgezeichnet
  - Der Angreifer kann nun die beiden Nachrichten wiederholen und die Zahlung mehrfach auslösen
  - Um das zu verhindern, werden die Nachrichten im Service gecached
  - Damit der Cache nicht unendlich groß sein muss, ist in Authent der Timestamp enthalten

- f) Warum ist Replay bei Principals (z.B. Alice) auch ohne Cache nicht möglich?
- Principals bekommen nur auf Anfrage vom AS oder TGS ein Ticket
  - Diese Anfragen sind immer durch Noncen geschützt

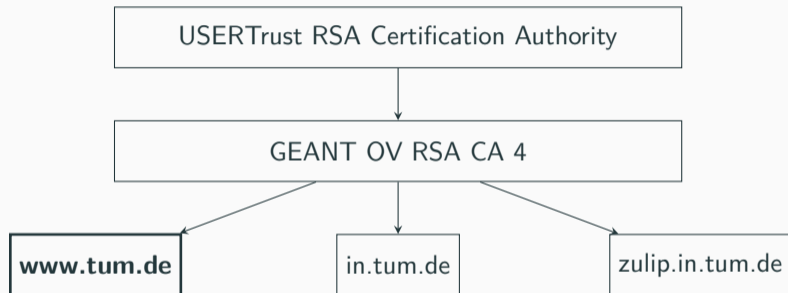
## Aufgabe 2

---



## Aufgabe 2a - PKI

Gegeben sei folgende Zertifikatskette:



- a) Welchen Zertifikaten aus der Zertifikatshierarchie muss ihr Browser vertrauen, wenn Sie `www.tum.de` öffnen?
- Nur *USERTrust RSA Certification Authority*, da es die Wurzel ist
  - In der Hierarchie wird allen Zertifikaten vertraut, die direkt oder indirekt von diesem **Wurzelzertifikat** signiert wurden

- b) Geben Sie alle Schritte an die zur Prüfung des Zertifikates erforderlich sind!
- 1) Stimmt das Subject (oder eins der Subject Alternative Names) mit der Webseitendomain überein?
  - 2) Ist das Zertifikat im Gültigkeitszeitraum?
  - 3) Sind die verwendeten Krypto-Primitive (wie z.B. das Hashingverfahren zum signieren) stark genug?
  - 4) Wird das Zertifikat für seinen Bestimmungszweck genutzt? (z.B. CA:TRUE oder CA:FALSE)
  - 5) Ist das Zertifikat zurückgerufen worden? Dies kann mittels CRL oder OCSP geprüft werden.
  - 6)  $Verify(Cert_{Sig}, C_{parent}^{pub}) \stackrel{?}{=} H(\text{Zertifikat})$
  - 7) Wird dem Zertifikat vertraut? (d.h. wird dem Zertifikat direkt vertraut, oder gibt es eine lückenlose Zertifikatskette, die in einem vertrauten Stammzertifikat endet?)

- c) Sie finden eine Webseite, deren HTTPS-Zertifikat mittels dem von `www.tum.de` signiert wurde. Würde ihr Browser dies akzeptieren?
- **NEIN**
  - Im TUM-Zertifikat ist die Flag `CA:FALSE` gesetzt