Das Kerberos-Protokoll

Eine Einführung in die Logik des Protokolls

Zu den Chemnitzer Linuxtagen 2012

von Mathias Feiler

(Kommunikations- Informations- und Medienzentrum der Universität Hohenheim)

Vortrag und Quasi-Workshop

- Vortrag
  - Vortragssprache ist Badisch (und Hosch-Schwäbisch)
  - Bei bedarf Übersetzung ins Hochdeutsche möglich
  - Dauer : Ca. 45 Minuten
- Optional: Theaterworkshop „Der Ur-Kerberos“
  - 5 Personen ca. 30 Minuten pro „Aufführung“
  - 2 Aufführungen simultan möglich.
- Legastheniker sind orthographisch fantasievoll!
  - Wer einen Fehler findet darf ihn behalten.

Die Gegebenheit

- Wir haben
  - Viele Server
  - Noch mehr Services
    - Filesystem (z.B. AFS)
    - Email
    - Web
    - DFÜ / RAS
    - Lernplattformen
    - WLAN
    - Cisco Router
- Wir suchen **eine** Anmeldemethode
  - Zentral steuerbar
  - Benutzerfreundlich
  - Nur ein Passwort
  - Für alle Services
  - Für alle Server
  - Sicher
  - Robust
  - Standardisiert

Wunschziele

- Eine Netzwerk-Aut**hent**isierung
  - In bzw. über offene Netze
  - Mit zuvor unbekannten Klienten
- Gegenseitiger Beweis der Identitäten
- Transistivität des Beweises für/gegen Dritte
  - Z.B. für Login-service, Email, Fileservice etc.
- Administrativer Aufwand muss gering sein
- Einmalige Anmeldung (Single sign on)
  - Für möglichst **alle** Services auf ein mal

Problem

- Netzwerke sind unsicher
  - Werden von jedem abgehört
  - Logisch: die Auth. muss auch via [VW]LAN funktionieren
- Die Klienten-Rechner sind vorab unbekannt
  - Kein 'pre shared key' möglich
- Authentisierung nur auf standardisierte Weise
  - Security statt obscurity
  - Installation von Standardsoftware muss genügen

Die Antwort : Kerberos

- Kerberos
  - Wird weitgehend unterstützt
  - Erfüllt obige Wunschliste
  - Kommt mit den oben genannten Problemen zurecht
  - Etymologie
    - Cerberus oder Zerberus
    - Der (dreiköpfige) Höllenhund
    - Grischische Sagenwelt
    - Wächter oder am Tor zur Unterwelt

Kerberos allgemein

- Kerberos ist ein Authentifizierungssystem
  - Erbringt den Beweis über die Identität eines Benutzers
    - Fälschungssicher
    - Befristet
- Kerberos ist ein „trusted third party“ System
    - Servicenehmer muss dem Kerberos bekannt sein
    - Servicegeber muss dem Kerberos bekannt sein

Über dieses Vertrauen erlangen beide ein gemeinsames Geheimnis,
das sie zur gesicherten privaten Kommunikation benutzen können.

authentisieren: rechtsgültig machen, glaubwürdig machen
authentifizieren: Echtheit bezeugen, beglaubigen

Begriffe

- Authentisierung / Authentifizierung
  - Ist er der, für den er sich ausgibt?
- Autorisierung
  - Darf er mit dem Ding tun, was er tun will?
- Realm
  - Das Reich eines Kerberos-Servers (Personen und Dienste)
- Prinzipal
  - Eine Identität in einem Kerberos-Realm (oft ein Mensch)
- Key (oder Schlüssel)
  - Parameter der (De-)Chiffrierung zur Ver- bzw. Entschlüsselung
- Ticket (oder gelegentlich Token)
  - Eine, in/mit dem Server-Key verschlüsselte Datenstruktur
  - Zeitlich eng begrenzter Ausweis (vergl. Fahrschein)

Was ist Authentifizierung

- Der Nachweis der Identität einer Entitiät
  - Eine Entität im Kerberos wird „Prinzipal“ genannt
    - Ein Prinzipal ist ein Kerberos-Teilnehmer
- Kerberos kennt die „mutual authentication“
  - Gegen- oder Wechselseitige Authentifizierung
    - Die Bank will wissen wer vor dem Automat steht
    - Der Kunde sollte wissen wollen, ob der Automat auch wirklich von der Bank ist
  - Optional, muss vom Klienten gefordert werden

Die Authentifizierung muss...

- Durch zentralen Service erfolgen
  - Da die Klientrechner vorab unbekannt sind
  - Um den Administrationsaufwand zu begrenzen
- Auf gehärteten Systemen erfolgen
  - Physikalisch sicher
  - DV-seitig sicher ;-(?!?
- Auf fälschungssichere Weise erfolgen
- Terminiert und ggf. verlängerbar sein
  - Eine erteilte Zugangsberechtigung muss nach einiger Zeit (Minuten bis Stunden) wieder verfallen.

Realm

- Administrative Einheit bezogen auf Kerberos
- Ist der Gültigkeitsbereich eines Kerberos
  - Realm-Name
    - Sollte eindeutig sein
    - Meist DNS-Domain in Großbuchstaben
  - Prinzipale (= Bevölkerung des Realms)
    - Benutzer (Accountnamen)
    - Digitale Services (Service-Name) oder Hosts (FQDN)
  - Realms können gekoppelt werden
    - „Cross realm authentication“ / „Vertrauensstellung“

Ein Prinzipal ist ...

- Ein Datensatz in der Kerberos-Realm-Datenbank
- Eine Identität im Kerberos

  Präsentiert genau einen Teilnehmer des Kerberos-Authentifizierungssystems eines Realms

  - Zweibeiner (Humanoide Kohlenstoffeinheit) z.B.:

    feiler@UNI-HOHENHEIM.DE

  - Digitaler Service oder Host, z.B.:

    HTTP/web.rz.uni-hohenheim.de@UNI-HOHENHEIM.DE

    host/mail.rz.uni-hohenheim.de@UNI-HOHENHEIM.DE

Ein Prinzipal hat ....

- Ein Prinzipal hat genau einen Namen[1] – keine Nr.
- Zu jedem Prinzipal gibt es verschiedene Keys
  - Erzeugt aus dem Passwort (oder Zufall)
  - Auf verschiedene Weisen vercryptet
    - 3DES, ARCFOUR, AES256, … (ggf. auch DES)
  - Zusatzinformationen / Metadaten / Attribute
    - End-Ablaufdatum (expiration) des Prinzipals
    - Gültigkeitsfrist eines Tickets / Verlängerbarkeit
    - ...

1) Im MIT-K5 mit LDAP-Backend gibt es auch Aliase – aber noch keine Admin-Schnittstelle dazu.

Verschlüsselung bei Kerberos

- Kerberos nutzt symmetrische Verschlüsselung
  - Für den „persönlichen“ Schlüssel (Key)
    - Für den initialen Identitätsnachweis zwischen dem Kerberos und einem Prinzipal
  - Als Sessionkey
    - Für die nachfolgenden Aktivitäten mit Dritten (z.B. Kontakt zwischen Mensch und Mailserver

- Eventuell asymmetrische Verschlüsselung mgl.

  Zum Identitätsnachweis, aber nur wenn der Kerberosserver hart mit einer CA verbunden ist.

Kerberos Sessionkey

- Wird vom Kerberos vergeben, der wird darum auch auch 'KDC'= „Key Distribution Center“ genannt
- Wird den Prinzipalen nur verschlüsselt übermittelt
  - Hier kommen die privaten Schlüssel zum Einsatz
- Temporärer Schlüssel für eine Sitzung
- Zur Kommunikation mit Dritten (auch mit dem TGS)
- Die Kenntnis des Sessionkeys zeigt dem Dienstanbieter (Service) und dem Nutzer (Klient) jeweils die Authentizität des Anderen an (bei „mutual auth.“)

Kerberos Ticket allgemein

- Man benötigt pro Service (Service-Prinzipal) ein Ticket
  - (Vergl. Busfahrschein, Kinoeintrittskarte, ...)
- Vom Kerberos (auf Verlangen) für den Klienten
  - Zusammen mit dem passendem Sessionkey
- Kann nicht vom Klienten entschlüsselt werden
- Kann nur vom Dienstanbieter entschlüsselt werden
- Dient als Ausweis für/gegen den Dienstanbieter
  - Nur zusammen mit dem passenden Authenticator gültig.

Was ist ein Ticket?

- Opake (verschlüsselte) Datenstruktur, enthält:
  - Klient-Prinzipal, für das das Ticket ausgestellt wurde
  - Server-Prinzipal, für das das Ticket ausgestellt wurde
  - Session-Key zur Kommunikation zwischen Prinzipalen
  - Uhrzeit, wann das Tickent ausgestellt wurde
  - Gültigkeit des Tickets (Lebensdauer)
  - … Weitere Informationen und attribute
- Verschlüsselt im privaten Key des Dienstanbieters
  - Dienstanbieter = Service
- Ist „nichts wert“ ohne passenden Authenticator

Authenticator

- Der Authenticator wird vom Klienten hergestellt
  - Enthält folgendes, verschlüsselt im Sessionkey
    - Systemzeit in (pseudo-) Mikrosekunden
    - Name des Prinzipals
    - .. Weiteres, z.B. zufällige initiale Sequenznummer

Anmerkung:

Den Sessionkey hat der Klient zusammen mit dem Ticket bekommen.

Der Sessionkey war dabei mit dem privaten Key des Klienten verschlüsselt.

Zeit

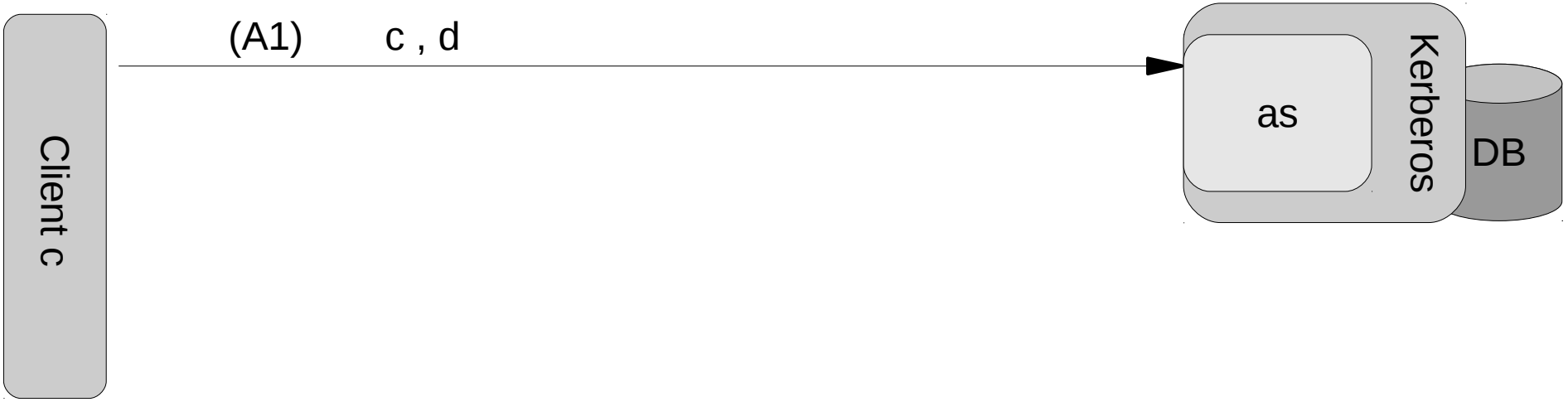
- Die Uhrzeit ist für Kerberos sehr wichtig
  - Um replay-Attacken zu erkennen
  - Um Paket-Duplikate zu identifizieren
  - Um abgelaufene Tickets zu erkennen
  - Um potentielle Fälschungen zu erkennen

- Üblicherweise tolerierte Zeitunterschiede:
  - 2 bis 5 Minuten zwischen Teilnehmern
  - Bei der „mutual authentication“ (wechselseitige Authentifizierung) muss der Dienstanbieter die (Mikrosekunden) genau selbe Zeit zurückgeben

Notation

c	Client (Prinzipal)
as	Authentication Service
tgs	Ticket Granting Service
d,app	Anbieter eines 3. Service, Dienstes oder Applikation, z.B. Notenabfrage
A(x)	Authenticator von x = x,Zeit
S(x,y)	Sessionkey für die geheime Kommunikation von x und y
T(x,y)	Ticket von x für y = x,y,S(x,y),Zeit,Lebensdauer,...
Kx	Privater Key von x
[ ]	Verschlüsselt
[ ]Kx	Verschlüsselt mit dem privaten Key von x
[ ]S(c,tgs)	Verschlüsselt mit dem Sessionkey von c und tgs

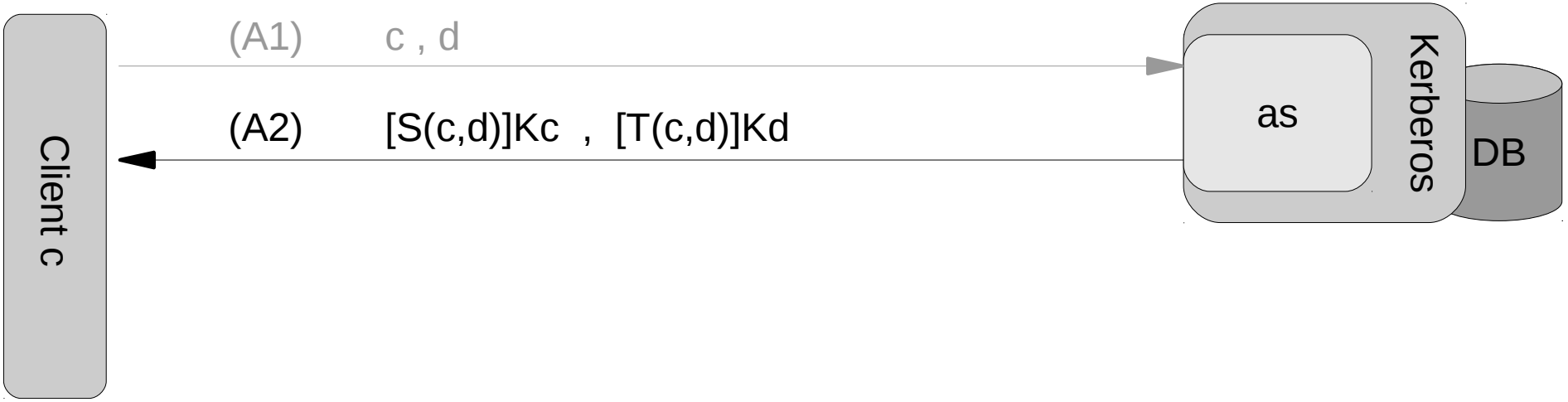
Schritt A1 : KRB_AS_REG



<u>Schritt A1:</u>
Client „c“ bittet den Authentisierunsservice (as) des Kerberos um Authentisierung für den Zugriff auf einen weiteren Servics „d“

Der AS des Kerberos sucht nun den zu „c“ und den zu „d“ passenden Key (Kc und Kd) aus seiner Datenbank heraus. Außerdem erfindet er einen Sessionkey „S(c,d)“ der zur Kommunikation zwischen „c“ und „d“ dienen wird.

Schritt A2 : KRB_AS_REP



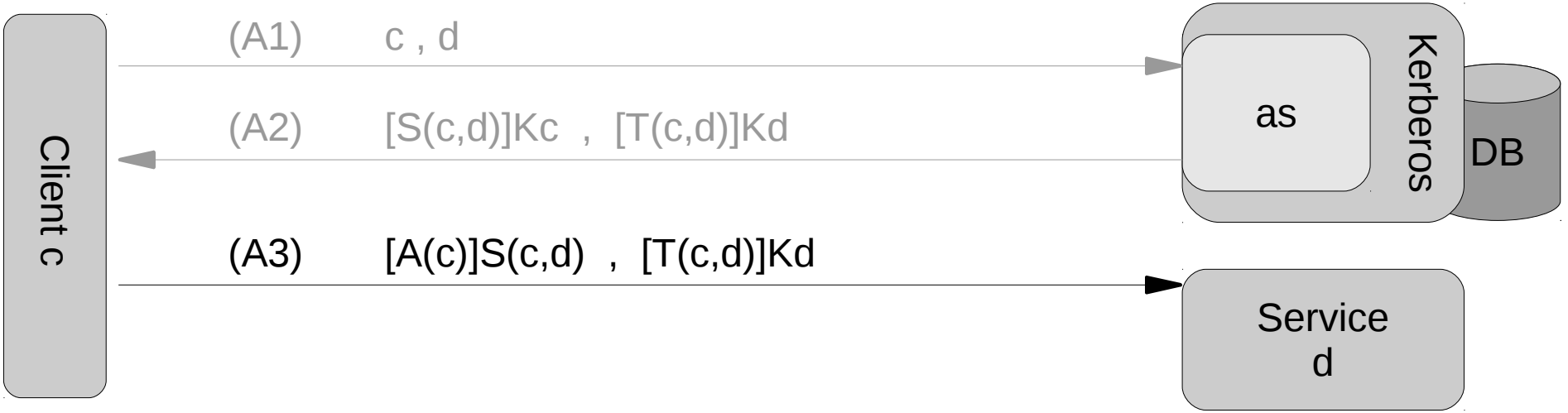
Schritt A2:

Kerberos (as) sendet

- Den Sessionkey verschlüsselt im Key des Klienten „Kc“
- Das Ticket „T(c,d)“ verschlüsselt im Key des Dienstanbieters „Kd“ an den anfragenden Klienten.

Der Client entpackt den Sessionkey und merkt ihn sich ebenso wie das verschlüsselte Ticket. T(c,d) = {c,d,S(c,d),Zeit,Lebensdauer,...}

Schritt A3 : KRB_AP_REQ



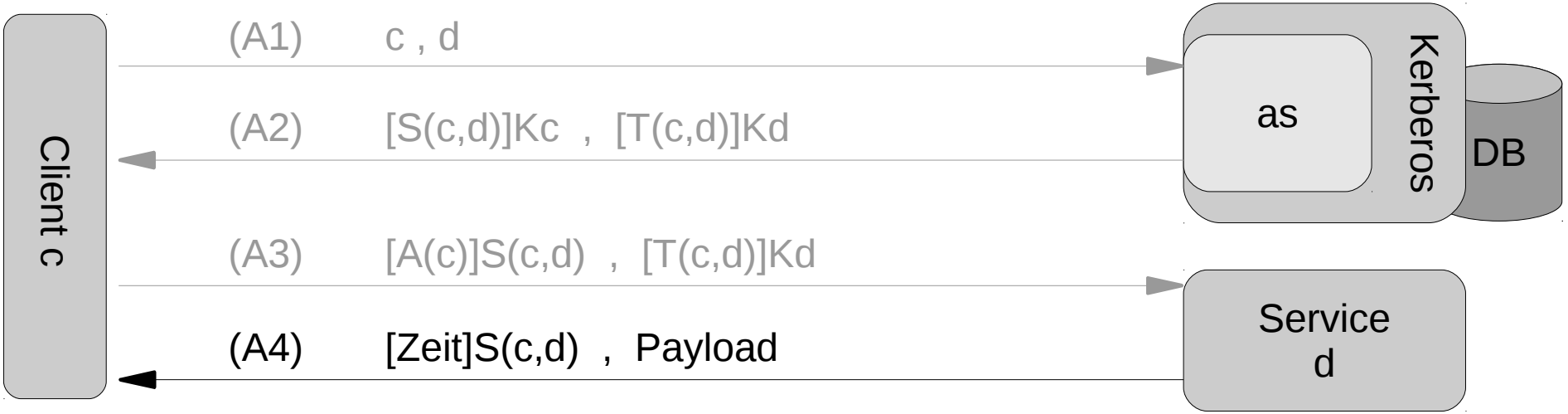
Schritt A3:

Der Client „c“ sendet folgendes zum Dienst „d“ :

- Einen im Sessionkey verschlüsselten Authenticator „A(c)“ = {c,zeit}
- Das eben erhaltene (im Key von „d“ verschlüsselte) Ticket.

Der Service „d“ Entschlüsselt das Ticket und erhält so den Sessionkey. Er versucht damit den Authenticator zu entschlüsseln. Danach wird noch die Gültigkeit überprüft

Schritt A4 :



Schritt A4:

Auf Veranlassung des Clienten findet die „Wechselseitige Authentisierung“statt. Dazu sendet „d“ die aus dem Authenticator bekannte Zeit verschlüsselt im Sessionkey zurück.

C prüft dies und weiß dadurch, dass d das Ticket entschlüsseln konnte. D muss folglich der sein, dessen Key in der selben Kerberosdatenbank hinterlegt ist wie der von c.

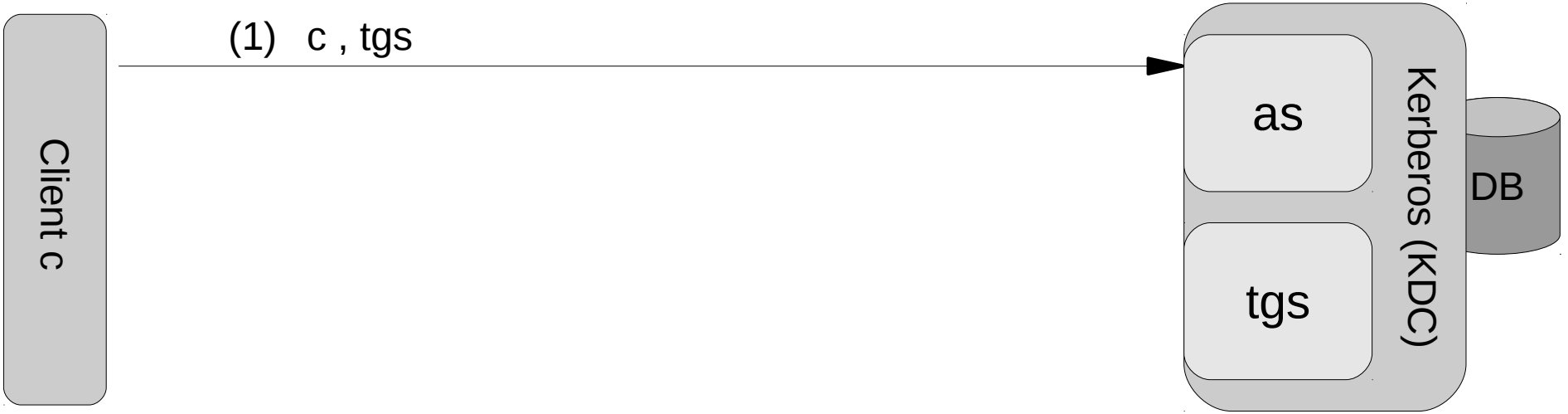
Fast gut!

- Methode ist logisch sicher
- Der Vorgang müsste pro Service/Server wiederholt werden
  - Mehrfache Eingabe des Passworts
    - Sicherheitskritisch (schielende Augen)
    - Unbequem
- Abhilfe
  - Einen Zwischenservice einführen (TGS)
    - Stellt Tickets für andere Services/Server aus
    - Ist selbst ein Kerberisierter Service
    - Läuft auf dem Kerberos server

Ticket Granting Service (TGS)

- Zugang mit primärem Ticket vom Kerberos
  - Dieses wird nun TGT (Ticket Granting Ticket) genannt
- Liefert Sessionkey + Ticket für andere Services
  - Natürlich erst nach eingehender Identitäts- und Plausibilitätsprüfung
- Besonderheit
  - Benutzt die selben Algorithmen wie der Kerberos AS
  - Braucht/Nutzt Zugang zur Kerberos-DB
  - Wird daher als integraler Teil des Kerberos implementiert

TGS-Schritt 1 : KRB_AS_REQ



<u>Schritt 1:</u> Wie gehabt:
„c“ bitted den Authentisierunsservice (as) des Kerberos um Credentials für den Zugriff auf den tgs

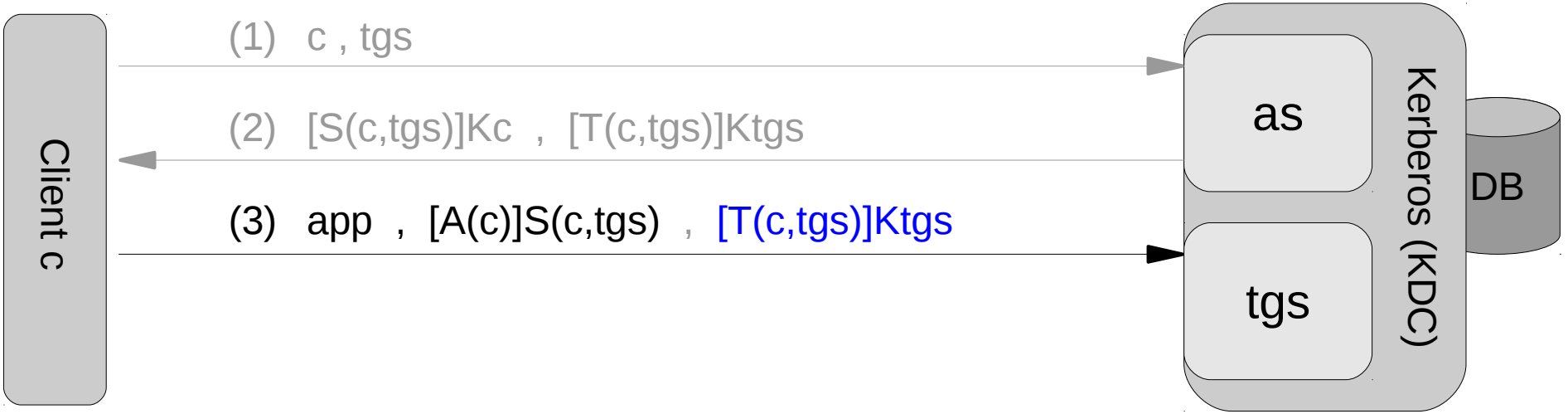
Der AS sucht die zu „c“ und „tgs“ passenden Keys (Kc und Ktgs) aus der Datenbank heraus. Außerdem kreiert er einen Sessionkey „S(c,tgs)“, der zur Kommunikation zwischen „c“ und „tgs“ dienen wird.

TGS-Schritt 2 : KRB_AS_REP

<u>Schritt 2:</u> Wie gehabt:
Kerberos (as) sendet an den Klienten:
- Den Sessionkey(c,tgs) verschlüsselt im Key des Clienten „Kc“
- Das im tgs-Key verschlüsselte Ticket GrantingTicket „T(c,tgs)“

C entpackt den Sessionkey und merkt ihn sich, ebenso wie das verschlüsselte Ticket.

TGS-Schritt 3 : KRB_TGS_REQ



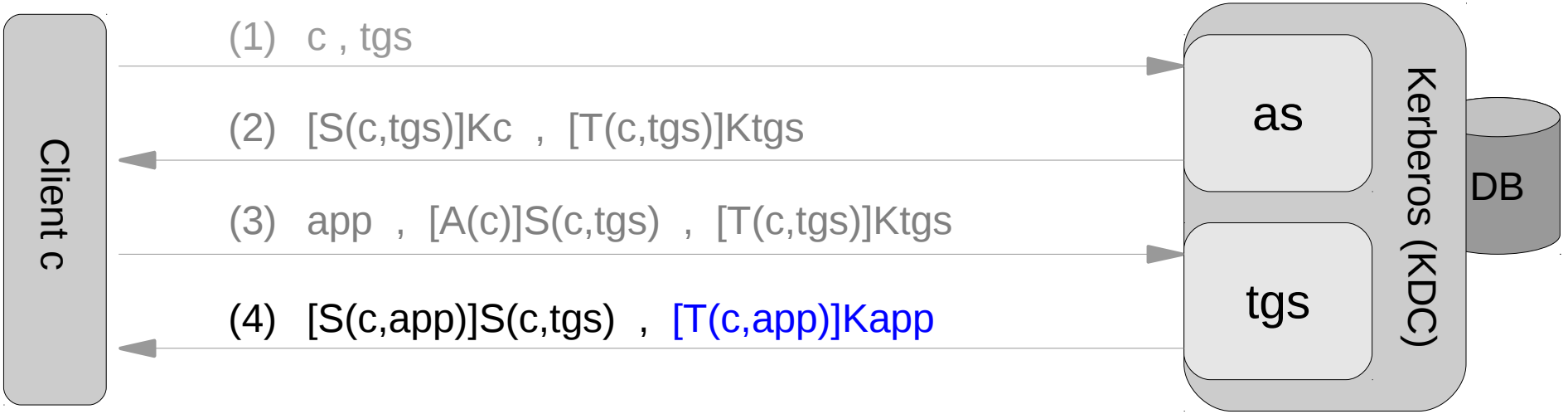
Schritt 3:

Der Client „c“ sendet folgendes zum TGS :

- Die Bitte um ein Ticket (+Sessionkey) für den Service „app“
- Einen im Sessionkey verschlüsselten Autenticator „A(c)“ = {c,zeit}
- Das eben erhaltene (im Key des TGS verschlüsselte) TGT.

Der TGS entschlüsselt das TGT und erhält so den Sessionkey. Er versucht damit den Authenticator zu entschlüsseln. Danach werden noch die Zeiten und Gültigkeit überprüft.

TGS-Schritt 4 : KRB_TGS_REP

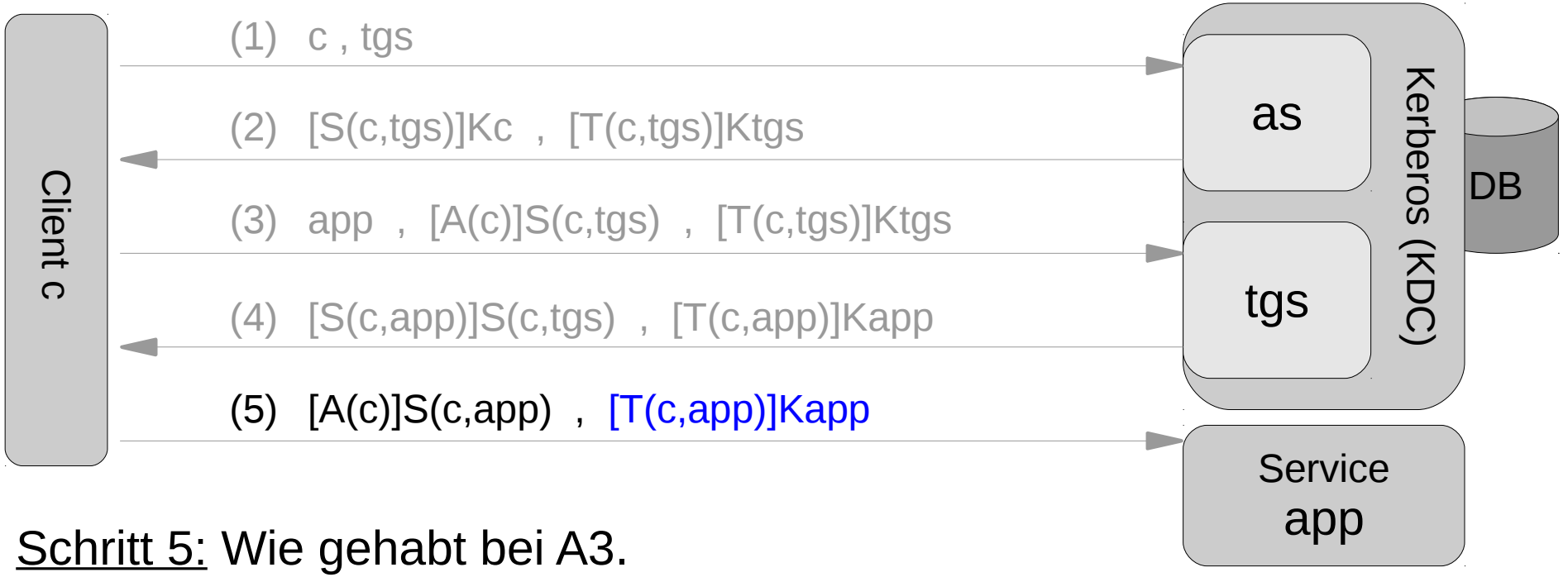


Schritt 4:

TGS sendet das Ticket für den Service „app“ und den dazugehörigen neuen Sessionkey S(c,app) verschlüsselt im Sessionkey(c,tgs)

C entschlüsselt den neuen Sessionkey(c,app) und merkt in sich zusammen mit dem dazugehörigen Ticket T(c,app) = {c, app, S(c,app), Zeit, Lebensdauer, … } .

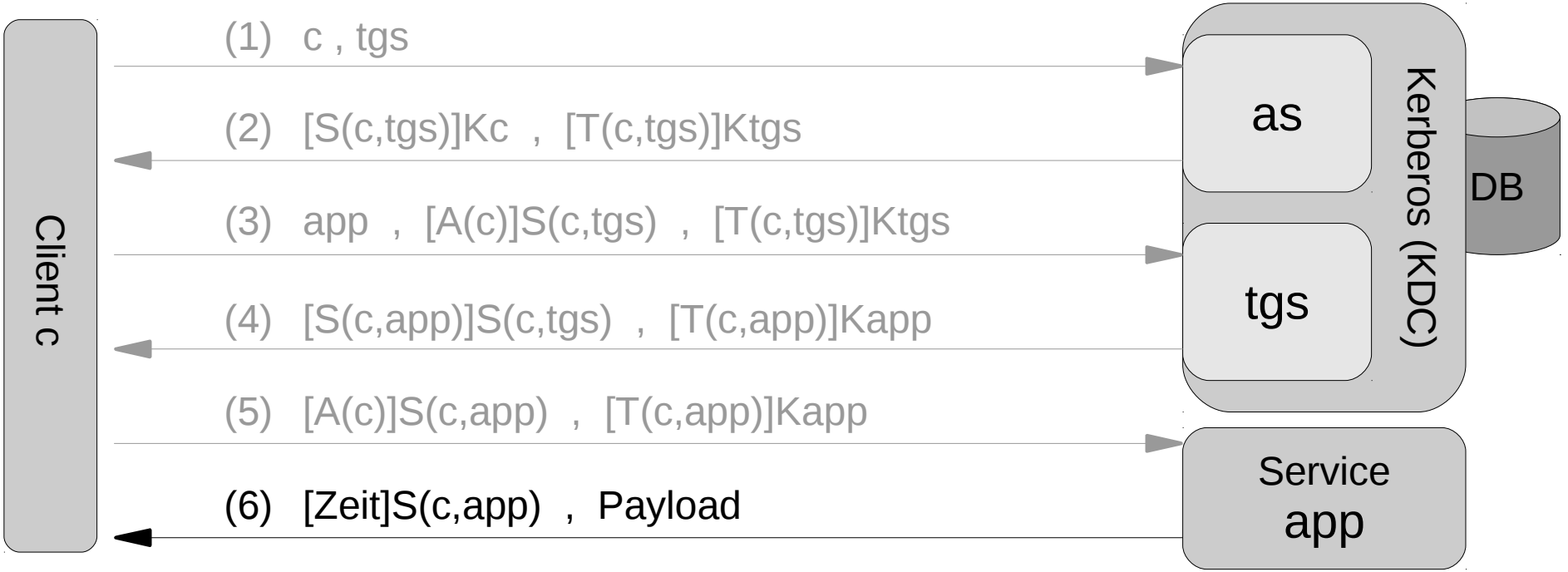
TGS-Schritt 5 : KRB_AP_REQ



Schritt 5: Wie gehabt bei A3.
Der Client „c“ sendet das Ticket und einen im passenden Sessionkey verschlüsselten Auhtenticator „A(c)“ = {c,zeit} zum Dienstanbieter „app“.

Der Service „app“ entschlüsselt das Ticket und erhält so den Sessionkey. Er entschlüsselt damit den Authenticator . Natürlich werden dabei allerlei Prüfungen durchgeführt.

TGS-Schritt 6 : KRB_AP_REP



Schritt 6: Wie gehabt bei A4.
Auf Veranlassung des Clienten findet die „Wechselseitige Authentisierung“ statt.
Dazu sendet app die aus dem Authenticator bekannte Zeit verschlüsselt im Sessionkey zurück.

So weiss c, dass ap das Ticket entschlüsseln konnte und ergo wirklich zum erwarteten Realm gehört.

TGT – Der Gewinn

- Die Schritte (1) & (2) mit der Passworteingabe sind nur ein mal notwendig
  - Man erhält daraus das TGT und den TGS-Sessionkey
- Schritt (3) & (4) können beliebig oft für verschiedene Services/Server automatisch wiederholt werden, da das TGT aus (1) & (2) vorliegt.
- Schritt (5) & (6) haben so bereits im oben vorgestellten Basisprotokoll automatisch funktioniert.

Jetzt ist es besser ...

- Methode ist immernoch lgisch sicher
- Benuter muss nur einmal das Passwort eingeben
  - Dafür bekommt er dann ein TGT + Sessionkey
  - Für (fast) lle Services des Realms, die der anzusprechen gedenkt
- Gegen das TGT bekommt der Bentzer beim TGS:
  - Tickets für (fast) „beliebe“ Server/Services
  - Tickets widerholt für den selben Server/Service
- Wir haben damit „Single Sign On“

… Aber

- Wirklich Jeder bekommt das TGT+Daten von Jedem
  - Um ein beobachtetes Passwort zu testen oder mit Versuchen zu vervollständigen.
- Jeder bekommt unkontrolliert viele TGT-Daten
  - Um das selbe Passwort auch gegen ander Prinzipale zu Testen?
  - Um ggf. offline zu knacken (Passwort erraten)

- Abhilfe : Pre-Authentication

Preauthentication 1

- Wurde in den originalen Kerber.* „vergessen“
  - Es handelt sich also um eine Erweiterung
- Reduziert Angriffsmöglichkeiten (erheblich)
- Ermöglicht das Hochzählen von Fehlversuchen
  - Ggf. (zeitweise) Sperrung des Prinzipals
    - Siehe bei MIT : „Lockout Policies“
- Implementierung
  - In (1) wird zusätzlich die in privaten Key von c verschlüsselte Zeit mitgeliefert
  - TGT bekommt nur, wer nachweislich den entsprechenden privaten Key von c kennt

Preauthentication 2

Auch mit Preauthentication kann versucht werden, den Key des TGS zu knacken. Dieser sollte darum öfter geändert werden.

Weitere Themen

- Anbindung / Kopplung mit PKI
  - Authentifizierung muss nicht mit einem Passwort erfolgen, sondern kann kann auch mit einem Zertifikat umgesetzt werden.
- Trust (Vertrauensstellung) mit anderen Kerber.*
  - Bei mehreren Realms z.B. in Sternform mgl.
  - auch mit AD möglich.
  - Durch Austausch der TGS-keys
- Credential-cache Manager (KCM)
  - Um mehrere Caches umschalten zu können.

Was Kerberos nicht ist / hat

- Authorisierung
    - Bank: Es ist Herr Feiler, aber darf er wirklich Geld abheben?
- Accounting
    - Wie viel Geld hat Herr Feiler den heute schon abgehoben?
- Kerberos kennt keine UID oder gar UUIDS
- Passwortänderung / Passwortmanagement
    - Nicht standardisierter separat kerberisierter Service
        - Glück: Heimdal und MIT sind sich einig.
    - Abhängig von der Implementation
        - MIT, Heimdal,GNU-shishi, AD,DCE
        - RFC 3244 für MS-Windows 2000

Kerberos Verstanden?

Aus dem Vortrag meines Kollegen :

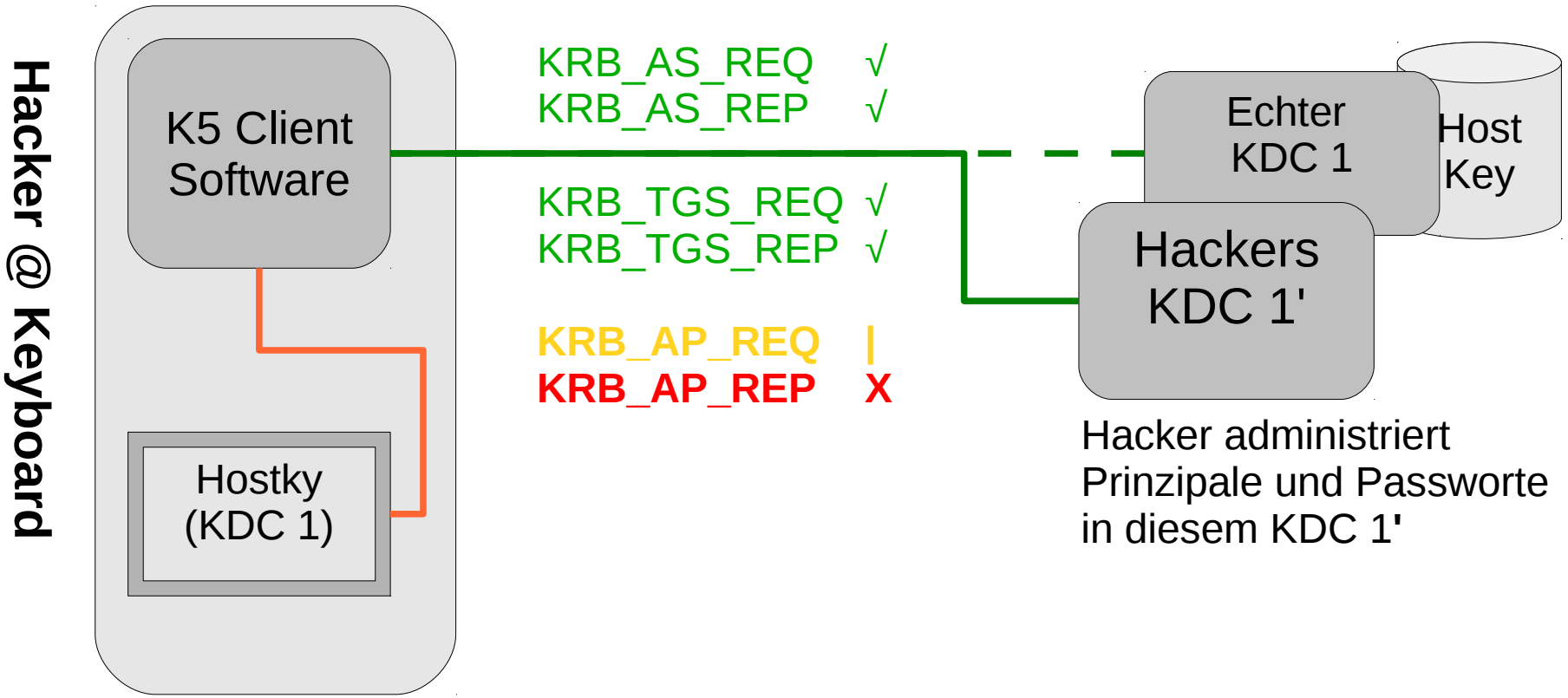
Reicht es für ein “kerberisiertes Login”, ein TGT anzufordern und zu prüfen, ob der im Schlüssel des Benutzers verschlüsselte Teil mittels des aus dem eingegebenen Passwort erzeugten Schlüssels entschlüsselt werden kann?

Anders gefragt :

Ist Folgendes für ein 'kerberisiertes' Login hinreichend?
* Der Clientenrechner fordert für einen User ein TGT an.
* Aus dem Passwort des Users wird der private Key erzeugt.
* Mit diesem Key kann der entsprechende Teil des KRB_AS_REP (der TGS-Sessionkey) entschlüsselt werden.

.... Und warum?

Antwort



Literarische Antwort : RFC 4120 Punkt 3.1.5 (Seite 28)

Literatur

- RFC4120

  Dokumentiert das Kerberos Protokoll

- Kerberos

  Mark Pröhl
  Kerberos
  Single-Sign-On in gemischten
  Linux/Windows-Umgebugen
  dpunkt.verlag ISBN 978-3-89864-444-0

- Distributed Services with OpenAFS

  Wolfgang A. Gehrke
  Springer Verlag ISBN 978-3-540-36633-1

Fragen ?

- ....

Vielen Dank

Für Ihre Aufmerksamkeit

Kerberos & AFS

Wie bindet man OpenAFS in ein Realm ein ?

(Big Picture)

Prinzipale für AFS anlegen

- AFS-Zellnamen festlegen
- Kerberos „noch“ so konfigurieren dass auch DES-Keys erzeugt werden
- Im KDC das Prinzipal für die AFS-Zelle anlegen

  … add afs/<cell.name>@REALM.NAME ...
- Im KDC ein AFS-Admin-prinzipal anlegen

  … add admin ...

AFS Keyfile extrahieren

- Heimdal:

```
cd /etc/openafs/server || cd /usr/afs/etc
kadmin -l ext_keytab -k AfsK5.kt afs/cell.nm
# ktutil -k AfsK5.kt list --keys
ktutil copy FILE:AfsK5.kt AFSKEYFILE:KeyFile
```

- MIT-K5

```
kadmin.local -q "ktadd -e des-cbc-crc:afs3 afs/cell.nm"
asetkey add <KVNR> /etc/krb5.keytab  afs/cell.nm
```

- Achtung:

  Die Key-Version-Nummer im KeyFile muss immer die neueste sein, die der KDC anzubieten hat.

AFS-Installation

- Afs installation wie gehabt
  - Bosserver -noauth
    - ptserver starten
    - Admin anlegen und in die Admin-Gruppe packen
      - pts createuser admin -id 1 ...
      - pts adduser admin system:administrators ...
    - vlserver anlegen und starten
    - fileserver anlegen und starten
    - Volume anlegen …
  - ....
  - bos setauth on

Für AFS authentisieren

- Allgemein
  - „kinit admin“ ....
  - „klist“
  - „aklog“ | „afslog“
  - „tokens“
  - … cd /afs/cell.name/... ....
  - „unlog ; tokens“
  - „kdestroy ; klist“
- Im Fall von gut konfiguriertem Heimdal kann 'a*log' entfallen.

Fragen ?

- ....

Vielen Dank

Für Ihre Aufmerksamkeit