Das Kerberos-Protokoll

Ein Beitrag zu den Chemnitzer Linuxtagen
März 2011

von Mathias Feiler

(Kommunikations-, Informations- und Medienzentrum der Universität Hohenheim)

Stand KW11/11

Vortrag und Workshop

- Vortrag
  - Vortragssprache ist Badisch (und Hoch-Schwäbisch)
  - Bei bedarf Übersetzung ins Hochdeutsche möglich
  - Dauer : ca 45-60 Minuten
- Workshop „der Ur-Kerberos“
  - 4 – 5 Personen * 45-60 min. pro “Aufführung”
  - Maximal 2 “Aufführung” gleichzeitig
  - Zwei Durchgänge
- Legastheniker sind orthographisch fantasievoll!
  - Wer einen Fehler findet darf in behalten.

Die Gegebenheit

- Wir haben
  - Viele Server
  - Viele Services
    - Email
    - Web
    - Filesystem
    - DFÜ / RAS
    - Lernplatform
    - WLAN
    - Cisco Router
    - ....
- Wir suchen **eine** Anmeldemethode
  - Zentral managebar
  - Benutzerfreudlich
  - Nur **ein** Passwort p.B.
  - Für alle Services
  - Für alle Server
  - Sicher
  - Robust
  - Standardisiert
  - ....

Wunsch-Ziele

- Eine Netzwerk-Aut**hent**isierung
  - In / über offene Netzwerke
  - mit zuvor unbekannten Klienten
- Gegenseitiger Beweis der Identitäten
- Transistivität des Beweises für Dritte
  - z.B. Für Login-service, Email, Fileservice ...
- Administrativer Aufwand muß gering sein
- Einmalige Anmeldung = “Single sign on”
  - Für möglichst **alle** Services

Problem

- Netzwerke sind unsicher
  - Werden von jedermann abgehört
  - Logisch: Auth. soll auch via VLAN etc. funktionieren
- Die Klientrechner sind vorab unbekannt
  - Kein 'preshared key' möglich
- Authentisierung nur auf standardisierte Weise
  - security statt obscurity
  - Installation von Standardsoftware muss genügen

Die Antwort: Kerberos

- Kerberos
  - Wird weitgehend unterstützt
  - Erfüllt die obige Wunschliste
  - Kommt mit den oben genannten Problemen zurecht

Historisches

- Wurde am MIT entwickelt
  - Teil des Project Athena
    - Kerberos sollte ursprünglich alles können: Authentisierug, Autorisierung und Accountig
- Ab ca. 1988 als Version 4 außerhalb des MIT.
  - V4 hat einige Nachteile und Schwächen (siehe unten)
- Ab 1993 als Version 5 (RFC 1510)
  - Zu locker spezifiziert → Merberos
- Seit 2005 gilt RFC 4120 für Kerberos V5

Kerberos allgemein

- Kerberos ist ein Authentisierungssystem
  - Erbringt den Beweis über die Idendität 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.

Begriffe

- Authentisierung
  - 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 Idendität in einem Kerberos-Realm . (Oft ein Mensch)
- Key (oder Schlüssel)
  - Parameter der (De-)Chiffrierung (Ver-und Entschlüsselung)
- Ticket (oder gelegentlich auch Token)
  - Eine im Serverkey verschlüsselte Datenstruktur
  - Zeitlich eng begrenzter Ausweis (vergl. Fahrschein)

Was ist Authentisierung

- Der Nachweis der Idendität eines Entity
  - Ein Entity auf kerberosisch heißt 'Prinzipal'
    - Prinzipal ist ein Kerberos-Teilnehmer – siehe unten
- Kerberos kennt die „mutual authentication“
  - Gegen- oder wechselseitige Authentisierung
    - Die Bank will wissen wer vor dem Automat steht
    - Der Kunde sollte wissen wollen, ob der Automat auch von der Bank ist.

Die Authentisierung muss...

- Durch zentralen Service erfolgen
  - Da die Klientenrechner vorab unbekannt sind.
  - Um den Administrationsaufwand zu begrenzen
- Auf gehärteten Systemen erfolgen
  - Physkalisch sicher
  - DV-seitig sicher ;-[?!?
- Auf fälschungssichere Weise erfolgen
- Terminiert und ggf. verlängerbar sein
  - Eine erteilte Zugangsberechtigung muss nach einigen Stunden wieder verfallen.

Autorisierung

- Klärung der Frage

  Darf der Akteur diese Aktion auf das Objekt anwenden?

  (Merke: Die Authentisierung sellt nur den 'Akteur' sicher)
- Nur als bzw. in Folge der Authentisierung
- Kann (bzw. sollte) dezentral geregelt werden
- Abhängig von Objekt und möglichen Aktionen
  - Werte i.A. unabhängig von der Authentisierung
  - Ist daher beim Objekt-Broker gut aufgehoben
- Nicht Teil des Kerberos

Realm

- Administrative Einheit bezogen auf Kerberos
- Ist der Gültigkeitsbereich eines Kerberos
  - Realm-name
    - Sollte eindeutig sein
    - meist DNS-Domain in Großbuchstaben
  - Prinzipale (= Befölkerung des Realms)
    - Benutzer (Accountnamen)
    - Digitale Services (Service-name) oder Host
- Realms können gekoppelt werden
  - Cross Realm Authentication / Vertrauensstellung

Ein Principal ist ...

- Ein Datensatz in der Kerberos-Realm-Datenbank
- Ein Principal ist eine Idendität im Kerberos

  Präsentiert genau einen Teilnehmer am Kerberos-Authentisierungssystem eines Realms

  - Zweibeiner (humanoide Kohlestoffeinheit) z.B.:

    `feiler@UNI-HOHENHEIM.DE`

  - digitaler Service oder Host, z.B.:

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

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

Ein Principal hat ...

- Ein Prinzipal hat genau einen Namen – keine Nr.
- Zu jedem Prinzipal gibt es verschiedene Keys
  - Erzeugt aus einem 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
  - ....

Idenditätsnachweis

- Idenditätsnachweis mittels Kerberos
  - Allgemein
    - Benutzung des gemeinsamen Key
    - Key ist privat (geheim , nur Kerberos und Klient bekannt)
  - Menschliche Prinzipale
    - Key wird min. aus Prinzipal und Passwort erzeugt
    - Passwort geht für die auth. nie über das Netz
    - Kein 'preshared key' jedoch 'preshared secret' (Passwort)
  - Elektronische Prinzipale (z.B. Fileserver)
    - 'preshared key'
    - Key wird vom Admin in einer Datei hinterlegt.

Verschlüsselungsarten allgemein

- Symetrische Verschlüsselung
  - Mit einem einzigen Schlüssel (Key) kann Ver- und Entschlüsselt werden.
- Asymetrische Verschlüsselung
  - Schlüssel treten hier immer paarweise auf. Wenn der eine Key zum Verschlüsseln benutzt wurde, kann das Resultat nur mit dem zugehörigen anderen Schlüssel wieder entschlüsselt werden.

Man spricht in diesem Zusammenhang auch vom privaten bzw. öffentlichen Schlüssel und im weiteren von Zertifikaten

Verschlüsselung bei Kerberos

- Kerberos nutzt symetrische Verschlüsselung
  - Für den “persönlichen” Key
    - für den initialen Idenditätsnachweis zwischen dem Kerberos und einem Prinzipal
  - Als Sessionkey
    - für die nachfolgenden Aktivitäten mit Dritten (z.B. Kontakt zwischen Mensch und Mailserver)

- Eventuell assymetrische Verschlüsselung mgl.
    - Zum Idenditätsnachsweis , aber nur wenn der Kerberosserver hart mit einer CA verbunden ist.

Kerberos Sessionkey

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

Kerberos Ticket allgemein

- Man benötigt pro Service-Prinzipal ein Ticket
- Vom Kerberos (auf Verlangen) für den Klienten
  - Zusammen mit dem passenden 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

Was ist ein Kerberos-Ticket?

- Opake (verschlüsselte) Datenstruktur, enthält:
  - Klient-Prinzipal, auf den das Ticket ausgestellt wurde
  - Server-Prinzipal, für den das Ticket ausgestellt wurde
  - Sessionkey zur Kommunikation zwischen Prinzipalen
  - Uhrzeit, wann das Ticket ausgestellt wurde
  - Gültigkeit des Tickets (Lebensdauer)
  - ...Weitere Informationen / Attribute
- Verschlüsselt im privaten Key des Dienstanbieters

  Dienstanbieter = Service
- Ist „nichts wert“ ohne passenden Authenticator

Authenticator

- Authenticator wird vom Klienten hergestellt
  - Enthält folgendes , verschlüsselt im Session-Key :
    - Systemzeit (in Mikrosekunden)
    - Name des Prinzipals
    - ...Weiteres, z.B. zufällige inititale Sequenznummer

  Den Session-key hat der Klient zusammen mit dem Ticket bekommen .

  Der Sessionkey war im privaten key des Klienten verschlüsselt.

Zeit

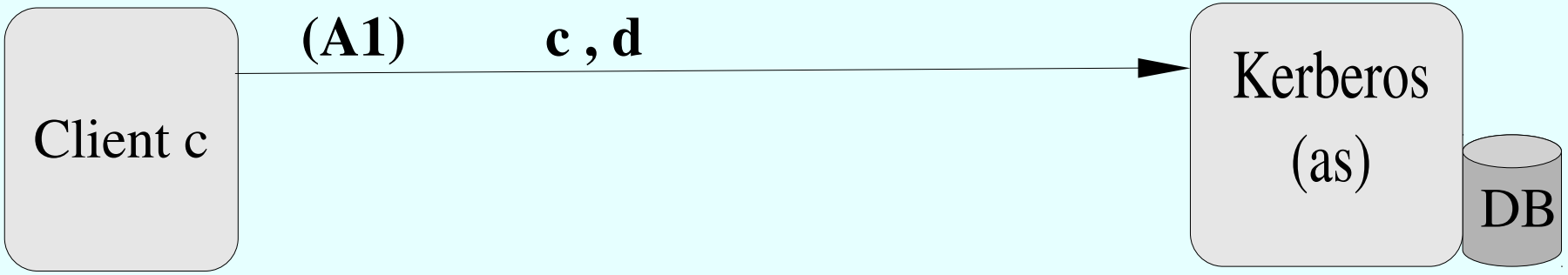
- Die Uhrzeit ist für Kerberos sehr wichtig
  - Um replay-Attacken zu erkennen
  - Um Paket-Duplikate zu idendifizieren
  - Um abgelaufene Tickets zu erkennen
  - Um potentielle Fälschungen zu erkennen

- Üblicherweise tollerierte Zeitunterschiede :
  - 2 bis 5 Minuten zwischen den Teilnehmern
  - Bei “mutual auth.” (wechselseitige Authentisierung) muss der Dienstanbieter die (mikrosekunden) genau selbe Zeit zurückgeben.

Notation

c	Client (Prinzipal)
as	Authentication Service
tgs	Ticket Granting Service
d,ap	Anbieter eines 3. Service z.B. Notenabfrage
A(x)	Authenticator von x = x,Zeit
S(x,y)	Sessionkey 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 im privaten Key von x
[ ]S(c,tgs)	Verschlüsselt im Sessionkey von c und tgs

Schritt A1 : KRB_AS_REQ

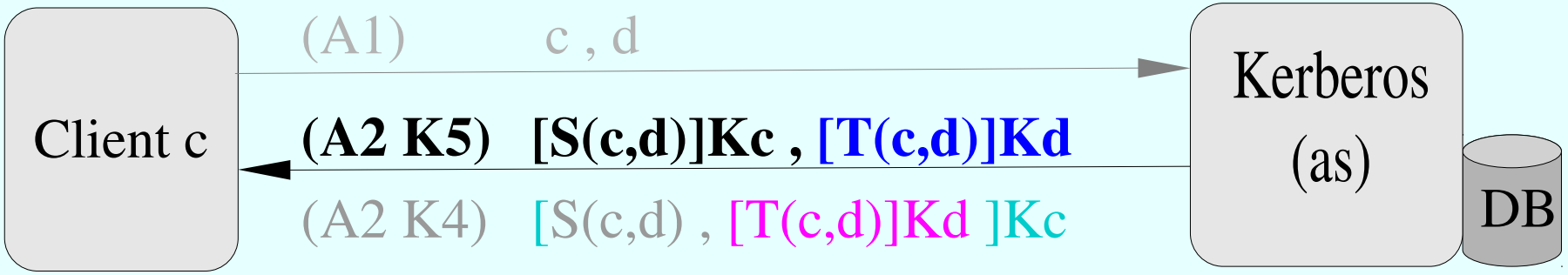


Schritt A1:

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



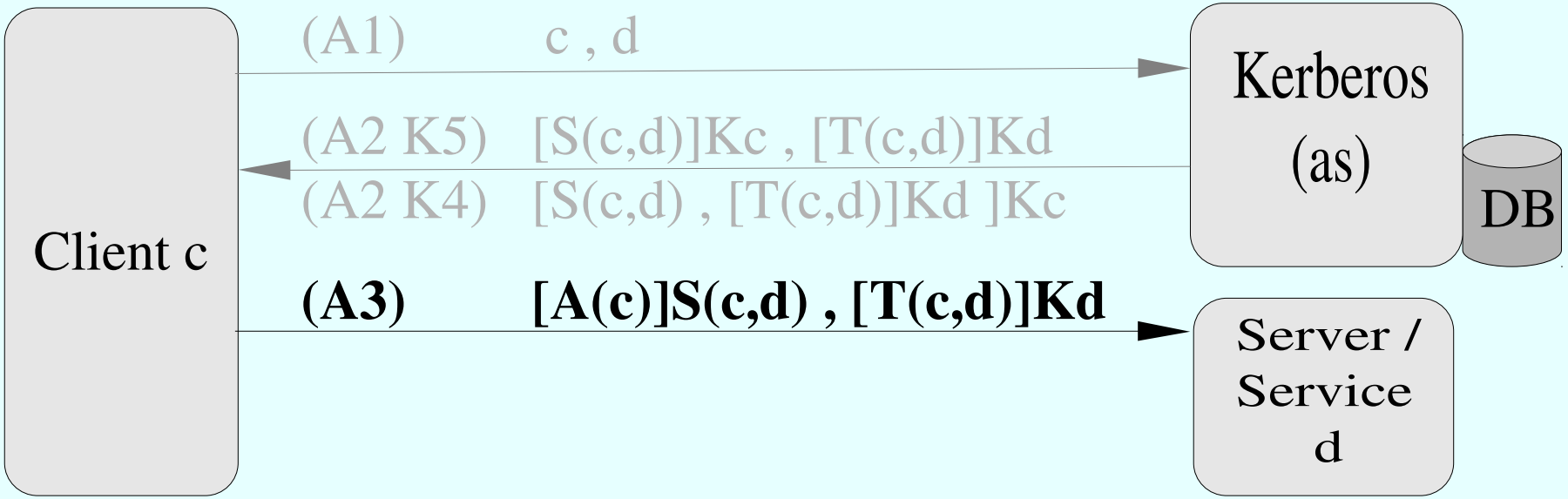
Schritt A2:

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



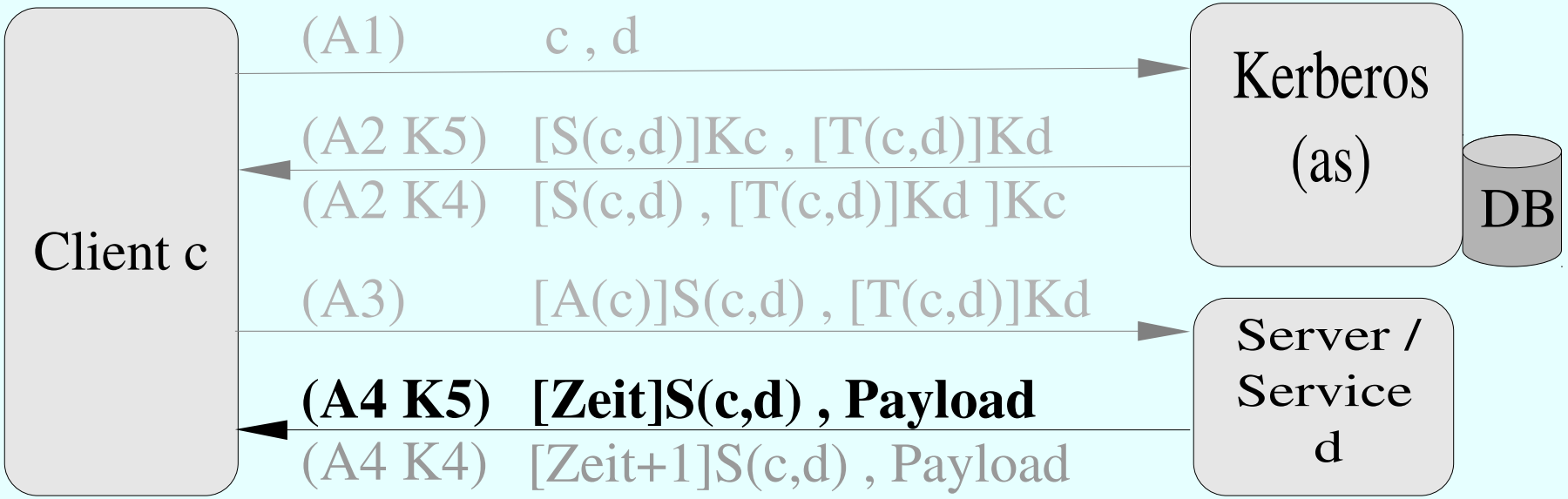
Schritt A3:

Der Client „c“ sendet folgendes zur Anwendung „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 : KRB_AP_REP



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
- Vorgang müßte pro Server wiederholt werden
  - Mehrfache Eingabe des Passwortes
    - Sicherheitskritisch (schielende Augen)
    - Unbequem (wunde Finger)
- Abhilfe
  - Einen Zwischen-Service einführen (TGS)
    - Stellt Tickets für andere Server/Services aus

Ticket Granting Service (TGS)

- Zugang mit primärem Ticket vom Kerberos
  - Dieses wird nun TGT (Ticket Granting Ticket) genannt
- Liefert Sessionkey + Ticket für andere Server
  - Natürlich erst nach eingehender Idenditäts- und Plausibilitätsprüfung
- Besonderheit
  - Benutzt die selben Algorithmen wie Kerberos
  - Nutzt Zugang zur Kerberos-Datenbank (sinnvoll!)
  - Wird daher als Teil des Kerberos implementiert

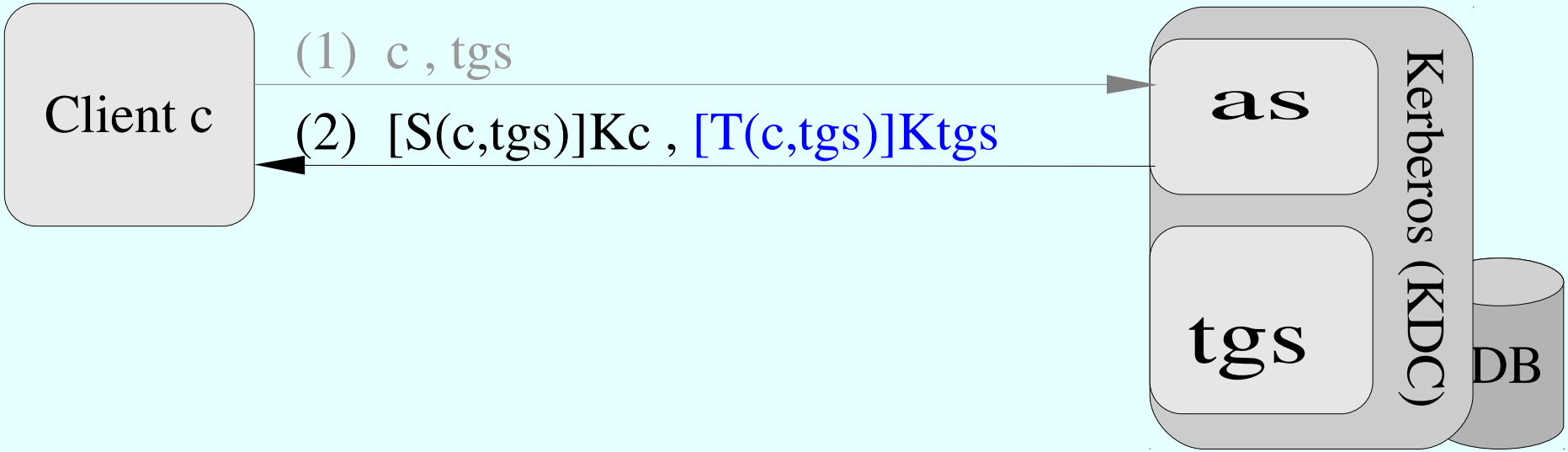
TGS-Schritt 1 : KRB_AS_REQ

<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



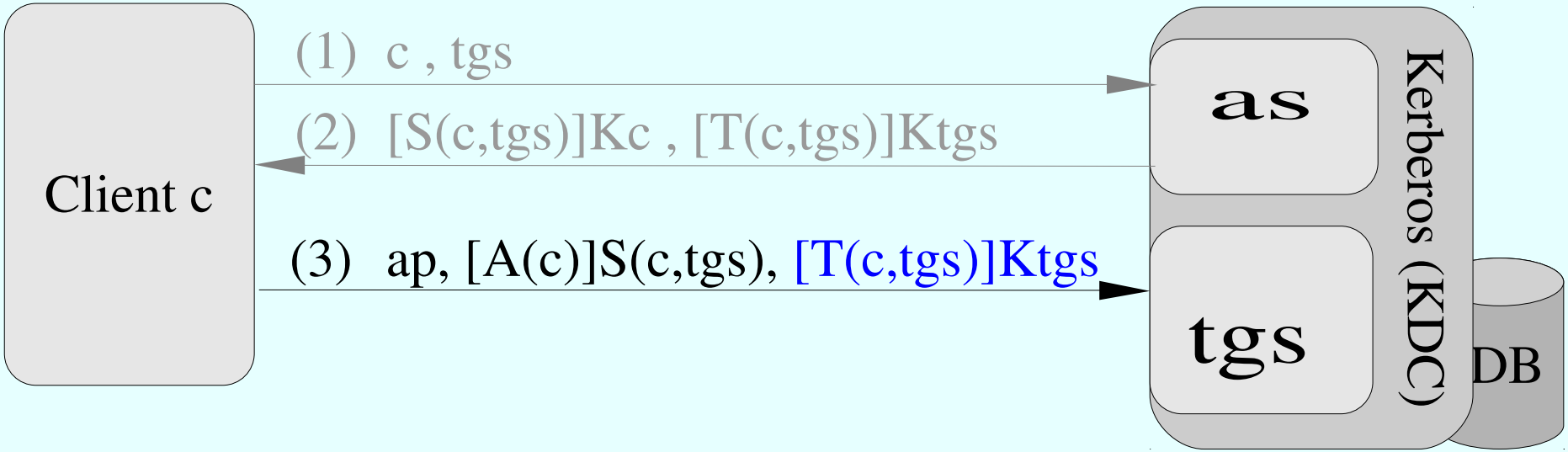
<u>Schritt 2:</u> Wie gehabt:

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



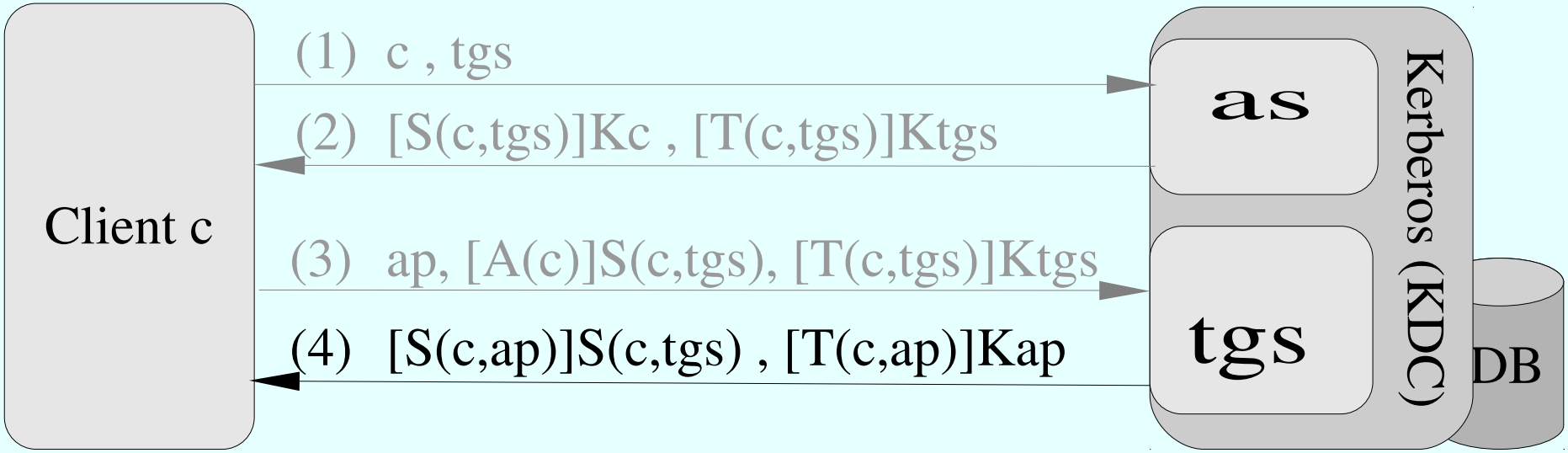
Schritt 3:

Der Client „c“ sendet folgendes zum TGS :

- Die Bitte um ein Ticket (+Sessionkey) für den Service „ap“
- 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

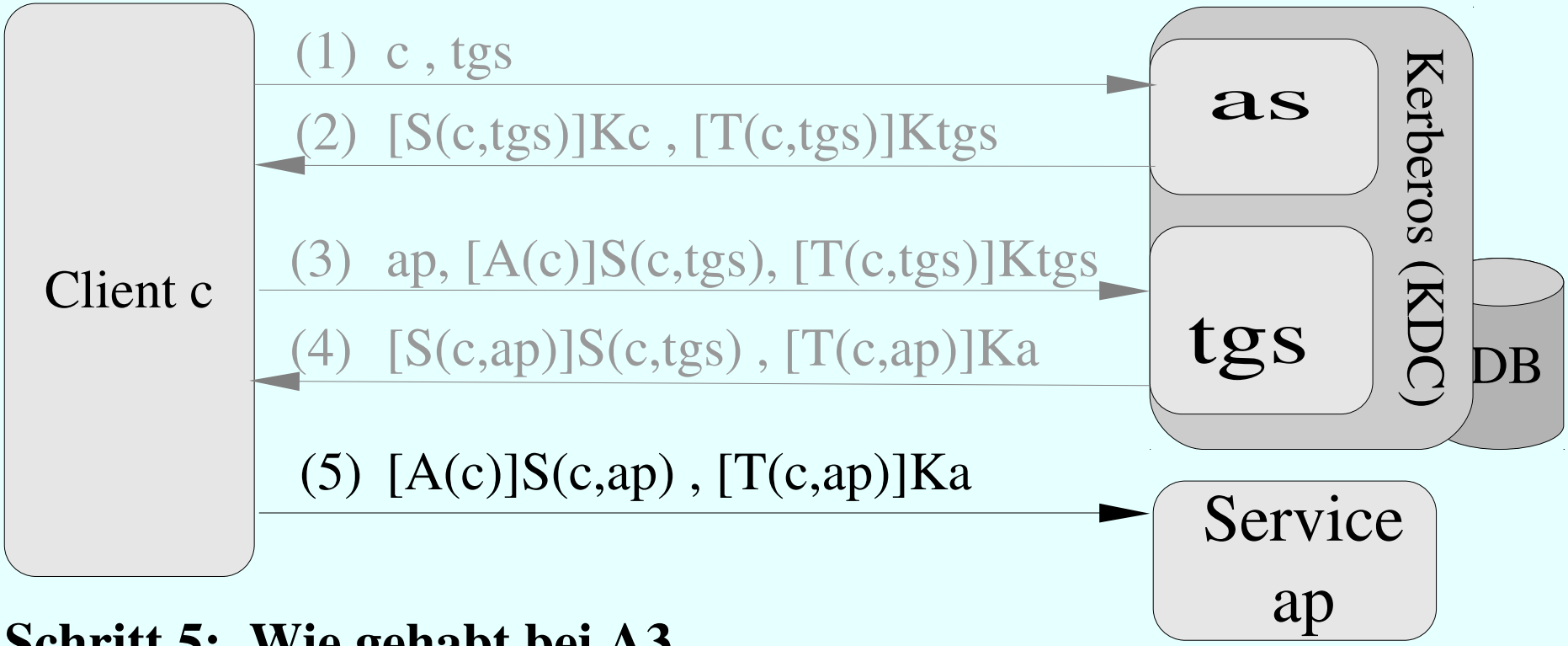


Schritt 4:

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

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

TGS-Schritt 5 : KRB_AP_REQ

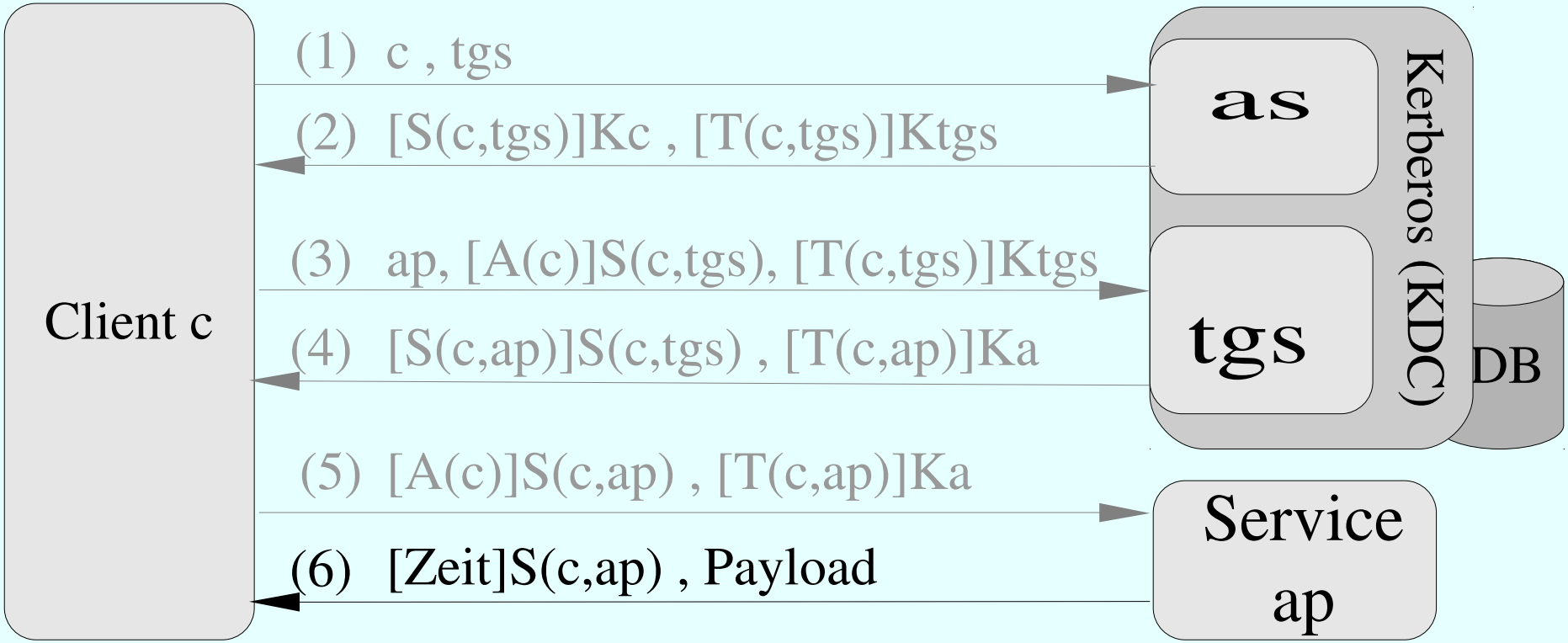


<u>Schritt 5:</u> Wie gehabt bei A3.

Der Client „c“ sendet das Ticket und einen im passenden Sessionkey verschlüsselten Auhtenticator „A(c)“ = {c,zeit} zum Dienstanbieter „ap“.

Der Service „ap“ 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



<u>Schritt 6:</u> Wie gehabt bei A4.

Auf Veranlassung des Clienten findet die „Wechselseitige Authentisierung“ statt. Dazu sendet ap 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) sind nur einmal notwendig
  - Man erhält daraus das TGT und den TGS-Sessionkey
- Schritt (3) und (4) können beliebig oft für verschiedene Services/Server automatisch wiederholt werden, da das TGT aus (1) und(2) vorliegt.
- Schritt (5) und (6) haben so bereits im Basisprotokoll automatisch funktioniert.

Jetzt ist es besser ...

- Methode ist immer noch logisch sicher
- Benutzer muss nur einmal das Passwort eingeben
  - Dafür bekommt er dann ein TGT+Sessionkey
  - Für (fast) alle Services des Realms, die er anzusprechen gedenkt
- Gegen das TGT bekommt der Benutzer beim TGS:
  - Tickets für (fast) “beliebige” Server / Service
  - Tickets wiederholt für den selben Server / Service
- Wir haben damit „single sign on“.

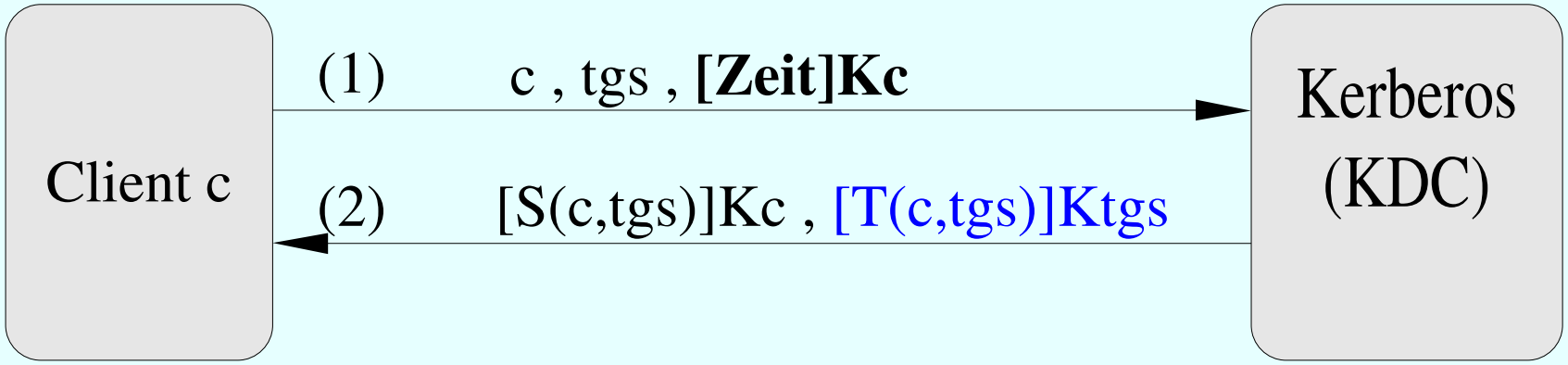
... Aber

- Wirklich Jeder bekommt 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 andere Prinzipal zu testen?
  - Um es 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
- Ermöglicht das hochzählen von Fehlversuchen
  - Ggf. (zeitweise) Sperrung des Prinzipals
- Implementierung
    - In (1) wird zusätzlich die im privaten Key von c verschlüsselte Zeit mitgesandt
    - TGT bekommt nur, wer nachweislich den entsprechenden privaten Key des Klienten kennt

Preauthentication 2



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

Kerberos V4 oder V5

- V4 ist eine “entlaufene Labor-Version”
- V5 ist Standardisiert(er)
  - Klienten kommen mit vorinstalliertem K5
  - Diverse Anwendungn haben Schnittstellen dazu
    - GSS-API , PAM , libkrb*, ..
  - “WinzigWeich” hat den “Merberos” vorgestellt:
    - K5-Unterart, die darauf achtet, irgendwelche Mehrdeutigkeiten im RFC innovativ zu benutzen.
    - Die anderen Kerberi haben gelernt damit umzugehen.

KerberosV4-Schmuddelecken (1)

- Verschlüsselung
  - Doppelte Verschlüsselung (gut, schlecht, sinnfrei?)
  - Klartext (Zeit) direkt verschlüsselt (gut für Hack!)
  - Nur DES (DataEncryptionStandard)
  - PCBC (Plain and Cipher Block Chaining)
    - Abart von CBC des DES
    - Fehlerhaft (Blöcke unerkannt austauschbar)
  - Fragliche CRC-Funktion (CRyptographic Cecksum)
    - Qualität nicht bestätigt – zweifelhaft?

KerberosV4-Schmuddelecken (2)

- Design-Blüten
  - Nur via TCP/IP möglich
  - Byteorder-Salat - “Receiver makes right”
    - Wieviel Byteorder-Codierungen sind für “int” vorstellbar?
  - Kein Ticket-forwarding vorgesehen
  - Prinzipal-Namen max. 39 Zeichen ohne '.'
    - Instanzen im K4 werden mit '.' getrennt.
    - FQDN-Problem
  - Inter-Realm-Authentication
    - Für jedes Realm ist dessen tgs-Key nötig (vollst. Vernetz.)
      - Ein Keywechsel ist damit aufwändig -> Security

V5 Defizite

- Ohne Erweiterungen pro Realm nur ein Ticket/TGT im Credential Cache
  - Entweder für “mf” XOR für “mf/admin”
  - Nur eines aus TGT , kdmin/admin , .. (→ Pr.Attribute)
- Ohne Erweiterungen nur ein Realm gleichzeitig
  - Problem beim Kopieren von Dateien zwischen Realms
    - Besonders, wenn man nicht dem “ganzen” Realm vertraut
- Administration nur Teilweise standardisiert
  - (potentielle) Protokollunterschiede
  - CLI-unterschiede

Weitere Themen

- Anbindung / Kopplung mit PKI
  - Authentiseirung muss nicht mit einem Passwort erfolgen, sondern kann auch mit einem Zertifikat umgesetzt werden.
- Trust (Vertrauensstellung) mit andern Kerberos
  - Bei mehreren Realms in z.B. Sternform mgl.
  - Imho. auch mit Active Directory Möglich.
  - durch Austausch eines TGS-Keys
- Credetial-cache Server (kcm)
  - Um mehrere Caches umschalten zu können.
- ....

Was Kerberos nicht ist / hat

- Authorisierung – siehe oben
- Accounting
- UID's oder gar UUIDS
- Passwortänderung / Management
  - Nich standardisierter separater kerberisierter Service
  - Abhängig von der Implementierung
    - MIT , Heimdal , GNU-Shishi , Active Directory, DCE
    - RFC 3244 für Microsoft 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?

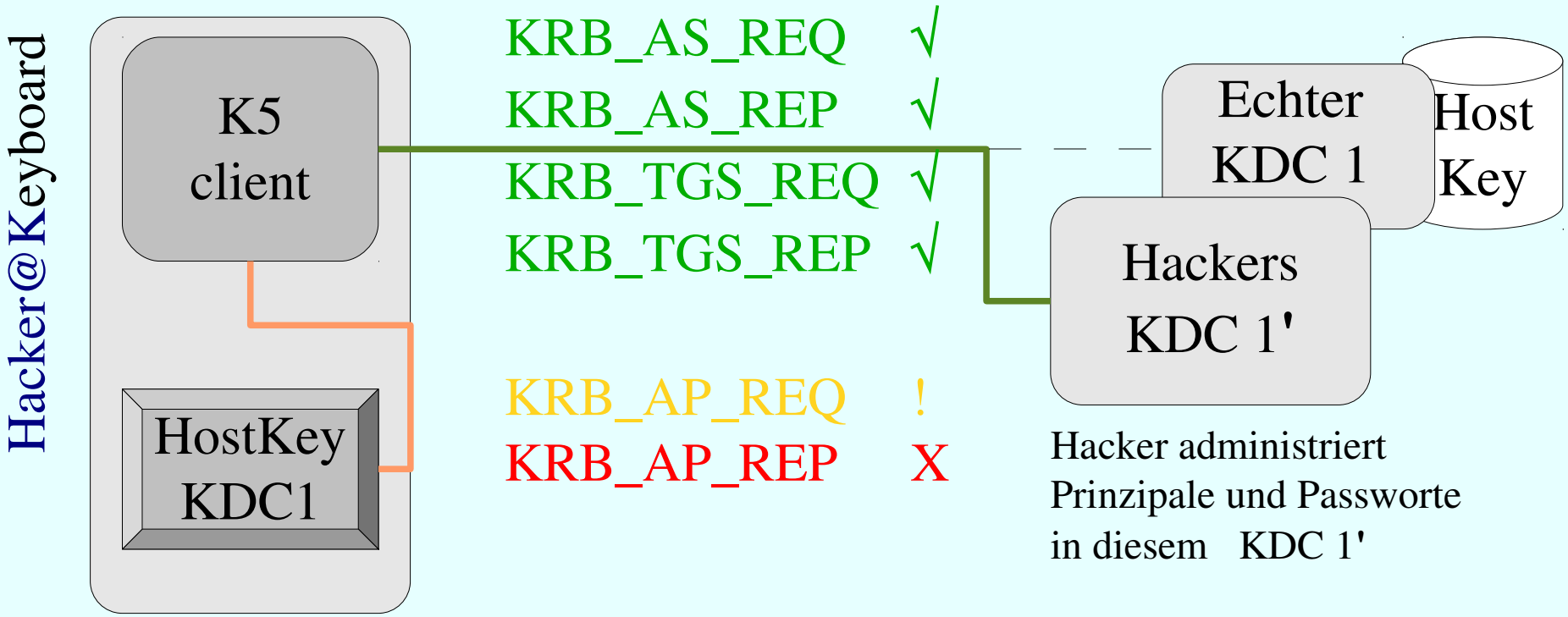
Anders gefragt :

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)

Fragen ?

- ....

Vielen Dank

Für die Aufmerksamkeit.