



# Authentifikation

---

Nils Hoier, Lutz Leitner, Benjamin Schleiner

Vortrag im Rahmen des Seminars 18.415 Sicherheit in vernetzten Systemen im WiSe 2002/2003  
Universität der Freien und Hansestadt Hamburg FB 18 - Informatik



# Übersicht

---

- Begriffsklärung
- Überblick über verschiedene Techniken
- Security Handshakes
- Kerberos V4
- Kerberos V5



## Was heisst "Authentifikation"?

---

- Authentifikation ist der Nachweis der Identität gegenüber dem Kommunikationspartner.
- Bei der Authentifikation wird der Anwender eines Systems auf seine tatsächliche Identität untersucht. So kann verhindert werden, dass ungerechtfertigt Dienste und Systemressourcen in Anspruch genommen werden können.



## Wie authentifizieren wir uns?

---

- Die Techniken, mit denen mich ein Computer (oder auch ein Mensch) identifizieren kann, lassen sich in drei Kategorien aufteilen:
  - Was ich weiß
  - Was ich habe
  - Was ich bin



## Passwörter

---

- Mit einem Passwort kann ich nachweisen, dass ich die Person bin, die ich vorgebe zu sein
- Ein Passwort passt in die Kategorie "Was ich weiß"



## Was heisst "Passwort"?

---

- Ein Passwort ist eine festgelegte Zeichenfolge zur Identifizierung und Legitimation eines Benutzers im Netzwerk
- Ursprung im Militär
- Wer das falsche Passwort nannte, wurde erschossen



## Passwortbasierte Authentifikation

---

- Das größte Problem ist das Abhören
- Passwörter können auch "geraten" werden
- On- vs. Off-Line Attack
- Sicherheitsfaktor Mensch!



## Probleme beim passwort- basierten Authentifizieren

---

- Ein Kollege kann Eingabe sehen
- Ein Eindringling kann ein gespeichertes Passwort lesen
- Das Passwort ist einfach zu raten
- Man kann das Passwort offline knacken
- Das System kann unbedienbar werden!
- User geben ihr Passwort weiter



## Online Passwort raten

---

- Oftmals sind Passwörter sehr einfach zu raten (sowohl Defaults als auch vom User).
- Vorsorge: Nach x Fehlversuchen das Konto sperren (Problem: Vandalismus)
- Vorsorge2: langsame Bearbeitung
- Vorsorge3: Logging ( & Nachricht)



## Offline Passwort raten

---

- Mit gegebenem Hash kann der Hacker ein Passwort offline raten
- Passwort raten, Hash berechnen, vergleichen
- Keine Beschränkung wie beim online raten
- Daher: Passwort muss besser sein!



## Weitere Risiken

---

- Ein Passwort für unterschiedliche Zwecke nutzen
- Trojaner
- Faulheit der User
- Fehlendes Verständnis der User für Sicherheit



## Sicherheitstokens

---

- Sind physikalisch vorhandene Gegenstände, die von einer Person mitgeführt werden können
- Passen in die "Was ich habe" Kategorie
- z.B. Schlüssel, Kreditkarte, Smart Card,...



## Adressbasierte Authentifikation

---

- Es werden keine Passwörter verschickt
- Zugriffsrechte hängen von der Absenderadresse ab
- Sobald ich die Adresse vorgaukeln kann und ahne, wo ich Zugriffsrechte habe, kann ich dort uneingeschränkt arbeiten



## Personenkontrolle durch Menschen

---

- Sicherheitsbeamte vor der Tür prüfen jeden, der in die Sicherheitszone will
- Nur möglich für
  - kleine Personengruppen
  - ähnliche Zugriffsrechte aller, die Zugang haben



## Biometrie

---

- Kategorie "Was ich bin"
- Man kann seine biometrischen Merkmale nicht ausleihen!
- Zur Zeit erhältliche Verfahren sind:
  - Retina Scanner
  - Fingerabdruck Leser
  - Gesichtserkennung



## Biometrie (2)

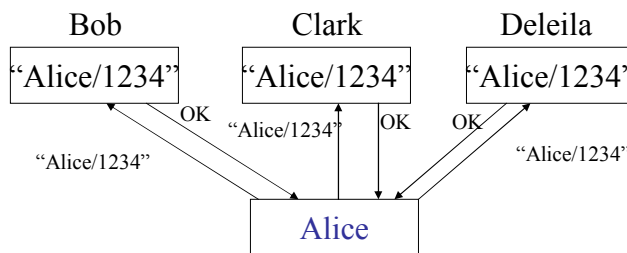
---

- Iris Scanner
- Handabdruck Leser
- Stimmenprofilanalyse
- Tastenanschlaganalyse
- Unterschriftenanalyse

## Wie erkennt ein Server Alice?

(1)

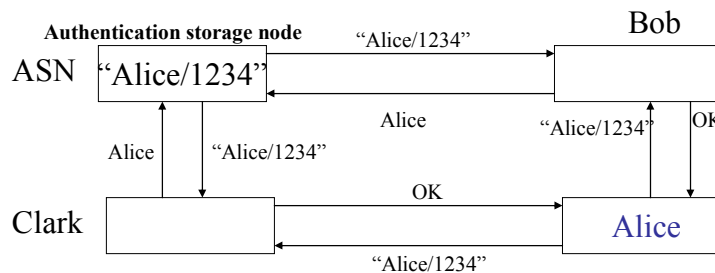
- Auf jedem Server, den Alice nutzen möchte, werden die Daten gespeichert



## Wie erkennt ein Server Alice?

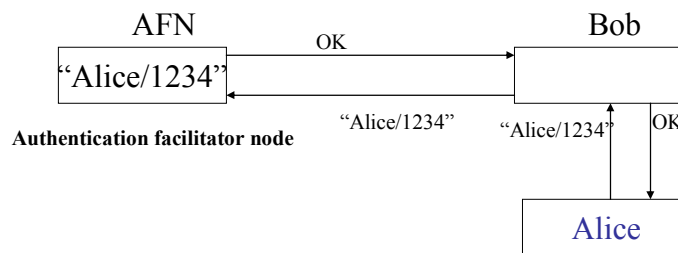
(2)

- Daten werden an einem zentralen Ort gespeichert und bei Bedarf weiter gegeben



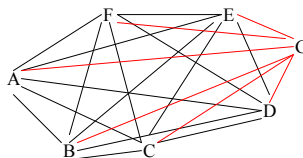
## Wie erkennt ein Server Alice? (3)

- Daten werden an einem zentralen Ort gespeichert und nur das Ergebnis einer Überprüfung wird übermittelt



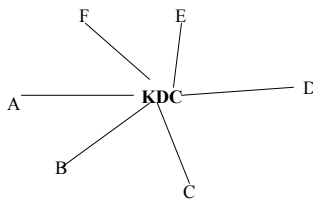
## Vertraute Vermittler

- Bei größer werdender Zahl kommunizierender Stellen steigt die Zahl der Schlüssel stark an



## KDC

- Darum Key Distribution Center
- Bei Gesprächsanfang werden Schlüssel vom KDC für beide Partner erstellt



## Security Handshakes

Protokolle und Gefahren



## Inhalt

---

- Entwurf von Grundlagen für sichere Protokolle

Beispiele:

1. Shared Secret
2. Public Keys
3. One-Way Public Key



## Entwurf von Grundlagen für sichere Protokolle

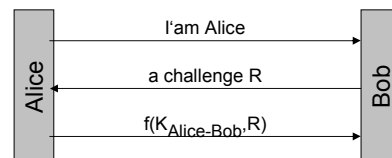
---

- Protokolle sollten angepasst werden an:
  - jeweilige Einsatzgebiete
  - verfügbare Ressourcen
- Leicht abgeänderte Protokolle können neue Sicherheitslücken hervorrufen
- Beim Entwurf neuer Protokolle sollte man sich von einem unsicheren Protokoll zum sicheren Protokoll vorarbeiten

## Login Only

- Alice sendet ihren Namen und ihr Passwort im Klartext über das Netzwerk zu Bob
- Bob überprüft Namen und Passwort
- Jede weitere Kommunikation erfolgt ohne Sicherheitsmassnahmen

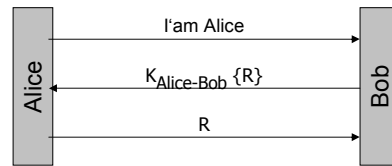
## 1. Shared Secret



- Gefahren:
  - Bob authentifiziert Alice, Alice authentifiziert Bob nicht.
  - Werden  $R$  und  $f(K_{\text{Alice-Bob}}, R)$  abgefangen, kann eine off-line password guessing attack erfolgen
  - Kann Bobs Datenbank gelesen werden, kann der Leser sich als Alice ausgeben
  - Jede weitere unverschlüsselte Kommunikation kann von Dritten abfangen werden
- Dieses einfache Protokoll bietet begrenzt Sicherheit, wenn nur eingeschränkt Ressourcen zur Verfügung stehen



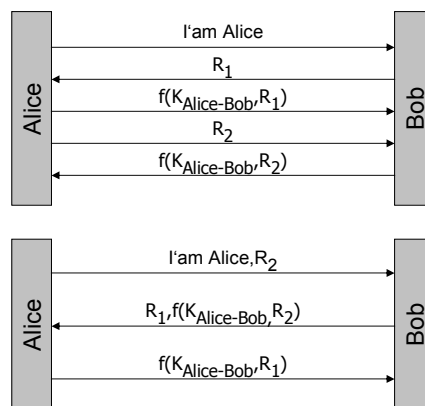
## 2. Shared Secret



- Gefahren:
  - Die selben Gefahren wie in der ersten Variante
  - Off-line attack auf  $K_{\text{Alice-Bob}} \{R\}$  möglich
- Um die Nachricht R zu entschlüsseln bedarf es „reversible cryptography“
- Bei begrenzter Lebensdauer von R, kann Alice Bob authentifizieren

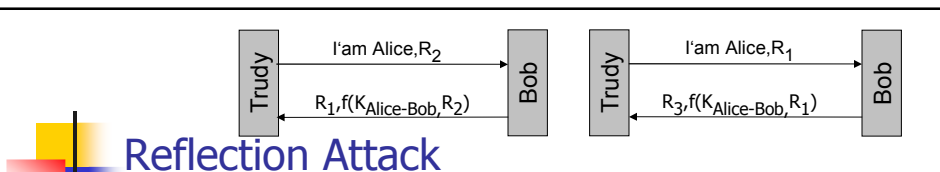
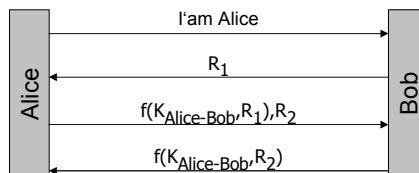


## Gegenseitige Authentifizierung



## Gegenseitige Authentifizierung

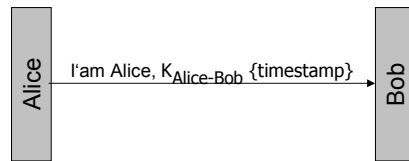
- Gefahren:
  - Da Bob „ $R_1, f(K_{\text{Alice-Bob}}, R_2)$ “ als Antwort sendet, ist eine off-line password guessing attack möglich
  - reflection attack
- Vermeiden der Gefahren durch senden einer weiteren Nachricht



- Dieser Angriff kann genutzt werden wenn entweder:
  - mehrere Verbindungen gleichzeitig zu Bob geöffnet werden können
  - $K_{\text{Alice-Bob}}$  für mehrere Server genutzt wird
- Zur Vermeidung sollten Alice und Bob entweder:
  - unterschiedliche Schlüssel benutzen
  - den Schlüssel für Bobs Authentifizierung von  $K_{\text{Alice-Bob}}$  ableiten
  - R darf von Alice nur als gerade, von Bob nur als ungerade Zahl gewählt werden



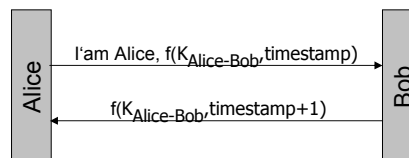
## Timestamp



- Gefahren:
  - Die Nachricht kann abfangen werden, um sie später erneut zu verwenden
  - Wird derselbe Schlüssel auf verschiedenen Servern verwendet, kann eine abgefangene Nachricht benutzt werden, um sich bei einem dritten Server zu authentifizieren
  - Kann bei Bob die Zeit zurückgesetzt werden, kann eine abgefangene Nachricht zur erneuten Authentifizierung genutzt werden
  - Basiert die Authentifizierung nur auf Zeit, kann sich keiner in dieses System einloggen
- Dieses Protokoll kann Protokolle ersetzen, bei dem die Passwörter im Klartext übertragen werden
- Das Protokoll ist effizienter und Bob muss sich kein R merken



## 2. Timestamp

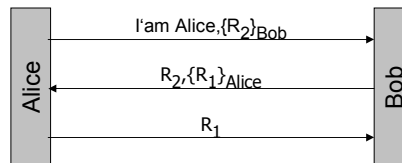


- Gefahren:
  - Die selben Gefahren wie in der ersten Variante
  - Die Nachricht  $f(K_{Alice-Bob}, timestamp+1)$  kann abfangen werden, um sie später erneut zu verwenden
- Dieses Protokoll kann request/response Protokolle ersetzen, bei dem die Passwörter im Klartext übertragen werden

## Verschlüsselung der Daten

- $K_{\text{Alice-Bob}}$  kann abgeändert werden und  $R$  verschlüsseln, um dann als Session Key benutzt zu werden
- Es sollte nicht  $K_{\text{Alice-Bob}} \{R\}$  oder  $K_{\text{Alice-Bob}} \{R+1\}$  als Session Key benutzt werden

## Public Keys



- Probleme:
  - Wie kann Alice Bobs public Key bekommen
  - Wie kann Alice ihren private Key bekommen
- Alice private Key liegt, mit ihrem Passwort gesichert, bei einem Verzeichnisdienst
- Bei dem selben Verzeichnisdienst liegt:
  - Bobs public Key mit Alice Passwort verschlüsselt
  - Ein Zertifikat für Bobs public Key, ausgestellt durch Alice



## Verschlüsselung der Daten

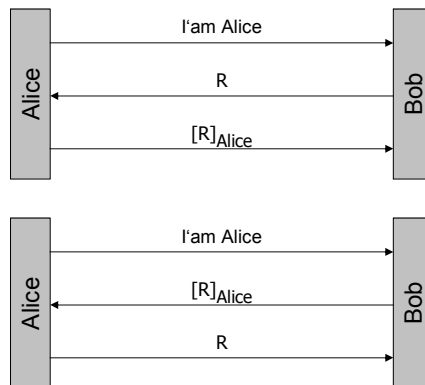
1. Alice wählt ein zufälliges  $R$  und sendet es, verschlüsselt mit Bobs public Key, an ihn
2. Alice kann zusätzlich das verschlüsselte  $R$  signieren
3. Alice wählt  $R_A$  und sendet es verschlüsselt an Bob. Bob wählt ein  $R_B$  und sendet es verschlüsselt an Alice
4. Alice und Bob führen einen Diffie-Hellmann Schlüsselaustausch durch.



## Diffie-Hellmann

1. Alice und Bob vereinbaren ein  $g$  und ein  $p$
2. Alice wählt ein zufälliges  $R_A$  und Bob ein zufälliges  $R_B$
3. Alice berechnet  $g^{R_A} \bmod p$  und schickt das Ergebnis signiert an Bob
4. Bob berechnet  $g^{R_B} \bmod p$  und schickt das Ergebnis signiert an Alice
5.  $g^{R_A R_B} \bmod p$  wird als Session Key benutzt

## One-Way Public Key



## One-Way Public Key

- Gefahren:
  - Jemand könnte sich als Bob ausgeben und Alice ein R schicken, um ihre Signatur für dieses R zu erhalten
  - Jemand könnte sich als Bob ausgeben und Alice eine für sie verschlüsselte Nachricht als R schicken, um als Antwort den Klartext zu erhalten
- Bei dieser Variante muss Bobs Datenbank nur vor unerlaubten Veränderungen geschützt sein



## Verschlüsselung der Daten

1. Alice wählt ein zufälliges  $R$  und sendet es, verschlüsselt mit Bobs public Key, an ihn
2. Alice und Bob führen einen Diffie-Hellmann Schlüsselaustausch durch, wobei nur Bob seinen Wert signiert



## Integrität/Verschlüsselung der Daten

- Es gibt keinen Algorithmus der mit einer Verschlüsselungsrunde oder einem Schlüssel sowohl die Integrität als auch die Verschlüsselung von Daten durchführt
- Das Abarbeiten der Nachrichten in der richtigen Reihenfolge sollte sichergestellt sein
- Es sollten regelmäßig neue session Keys vereinbart werden durch:
  - Verschlüsselung des neuen Schlüssels unter dem alten Schlüssel
  - Durchführung eines Diffie-Hellmann Schlüsselaustausch

## Überblick Kerberos

- Einleitung
- Motivation
- Systemarchitektur/Begrifflichkeiten
- Kerberos V4
- Kerberos V5
- Allg. Systemschwächen

## Einleitung (1)

- Kerberos ist ein schlüsselbasierender 3rd-Party Authentisierungsmechanismus  
→ Netzwerk Authentisierungs-Protokoll
- Der Name stammt aus der griechischen Mythologie → (meist) dreiköpfiger Höllenhund zum Eingang der Unterwelt. Gewährt Eingang, aber lässt niemanden die Rückkehr zu.





## Einleitung (2)

- Am MIT (*Massachusetts Institute of Technology*)  
1983 im Rahmen des Projektes „Athena“ entwickelt
  - Ziel von „Athena“ war die Implementation einer Client-Server-Landschaft grafischer Workstations im akademischen Umfeld
- basiert ursprünglich auf einem Entwurf von S. Miller und C. Neuman
  - Erweiterung des Needham-Schroeder Protokolls mit geheimen Schlüsseln
- Erste freie Implementierung 1987/88 mit V4,  
bis V3 nur MIT-interne Entwicklungsversionen
- V4 bis 1992 weiterentwickelt (PatchLevel 10)



## Einleitung (3)

- Bereits 1989 Beginn der Spezifikation von V5,  
Sept. `93 Endfassung von RFC 1510
- als OpenSource und in kommerziellen Produkten verfügbar  
MIT-Implementationen für Unix, Solaris, SunOS, Linux...  
Domänenauthentifikation in Windows 2000
- aktuelles Release ist 1.2.6
- Letzte Bugmeldung:  
[MITKRB5-SA-2002-002](#): [updated 2002-10-25] Buffer overflow  
in kadmind4  
Remote user can gain root access to KDC host.



## Motivation

- Die innere Sicherheit eines Netzes ist ebenso relevant wie die Gefahr von Außen → Standpunkt: Verbindungen sind unsicher (...die Rechner selber jedoch schon)
- Trennung Authentisierung und Autorisierung  
konventionell: keine Trennung, d.h. jeder Dienst hat seine eigene Authentisierung und somit jeweils ein eigenes Passwort  
Ziel: Authentisierung einmalig zentral, Autorisierung dezentral

*Zentraler Nutzen von Kerberos:*

→ Reduktion der Übermittlung von Passwörtern, somit Gefahrenminimierung dieser speziellen Angriffsmöglichkeit

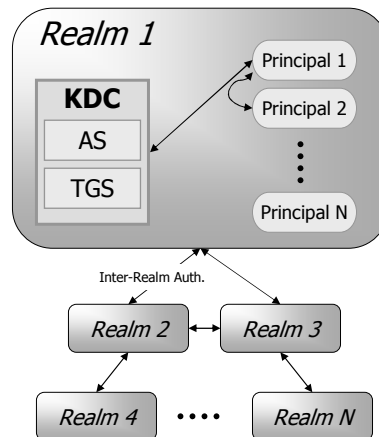


## Designaspekte

- V4/5 auf Ebene der Anwendungsschicht, spezielle Anpassung an die nutzenden Anwendungen nötig → Kerberizing
- V4 schneller/einfacher und weiter verbreitet
  - Nur TCP/IP
  - Verschlüsselung nur per DES (PCBC Plaintext Cipher Block Changing)
  - Zu V5 nicht aufwärtskompatibel
- V5 größere Funktionalität, aber konzeptionell gleich
  - Nicht auf TCP/IP beschränkt
  - Beliebige Verschlüsselungsmodule
  - Abwärtskompatibel zu V4

## Systemarchitektur

- Die Komponenten (3 Köpfe) beider Kerberos Versionen:
- **Realm**
- **Principal**
- **KDC (Key Distribution Center)**
  - **AS (Authentication Server)**
  - **TGS (Ticket Granting Service)**



## Realm

- Realms sind Domänen zur besseren Administration und zur Repräsentation von Organisationen
- vertrauen untereinander nur eingeschränkt
- enthält ohne weiteres 100000 Principals
- wird durch einen eindeutigen Namen identifiziert
  - in V4 durch 40-Zeichen String (DNS)
  - in V5 hierarchische Benennungsstruktur nach X.500 bzw. DNS  
→ nützlich bei Interrealm-Authentifikation (→)

Principals unterschiedlicher Realms führen die Authentifikation indirekt über Realm-Principal Beziehungen durch:

V4: muss (mind.) einer auch Principals des anderen sein

V5: können auch indirekt über andere Realm-Principals verzweigen



## Principals

- in Kerberos: Benutzer, Client, Clientprogramme/-prozesse  
...alle im Netzwerk miteinander kommunizierenden Objekte die Kerberos benutzen, jedoch nicht Komponenten von Kerberos
- Principals haben einen (realmweit) eindeutigen Bezeichner und gehören nur einem Realm an
- Bezeichner dreiteilig:
  - Name
  - Instanz (nur in V4; in V5 gliedert sich Name in mehrere Strings auf, sodass auch die Instanzbenennung weiterverwendet werden kann)
  - Realm

```
name [/instance]@REALM  
donald/admin@pentagon.mil  
donald@pentagon.mil
```



## KDC (*Key Distribution Center*) (1)

### Allgemein:

- Trusted Third Party: vertrauensvoller Vermittler, der hinzugezogen wird um die Echtheit der Kommunikationspartner zu verifizieren
- Jedes Prinzipal hat ein Geheimnis, das ihn gegenüber der TTP authentifiziert, da nur die TTP das Geheimnis mit ihm teilt  
→TTP kennt die Geheimnisse aller Principals
- Zwei Prinzipals authentifizieren sich somit indirekt über die TTP

### Übertragung nach Kerberos:

- Das KDC stellt die TTP dar
- In jeder Realm gibt es ein KDC (optional mehrere...)
- Principals vertrauen nur ihren Stamm-KDC



## KDC (*Key Distribution Center*) (2)

- **KDC** kennt Namen, Schlüssel (Master Key), (optionale) Attribute und administrative Daten all seiner Realm-Principals
  - Master Keys (abgeleitet aus dem Passwort) verschlüsselt mit dem Master KDC Key mit DES (V4) bzw. beliebige Methode (V5)  
→ zusätzliche Absicherung zwingend notwendig
- **KDC** unterteilt in:
  - **AS (Authentication Server)**
    - vergibt das **TGT (Ticket Granting Ticket)** als Basis für die Anmeldung an den **TGS**.
    - **AS** wird nur einmal während der Session beim ersten Login kontaktiert
  - **TGS (Ticket Granting Service)**
    - Kann nur mit passenden TGT vom AS verwendet werden
    - Vergibt an den Principal die Tickets für die gewünschten Kommunikationspartner
- Funktionale Teilung → einmaliger Benutzerlogin

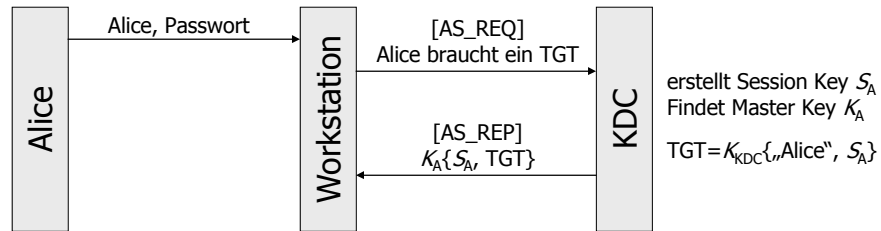


## Replizierte KDC's

- Realm kann mehr als ein KDC enthalten
- diese sind zueinander äquivalent
  - inhaltlich identisch: Teilen den gleichen Master KDC Key, gleiche Principals & deren Master Keys
- Unterscheidung nach Master und Slave:
  - Der Master verwaltet die Master-Copy
  - Master ist Zentralpunkt für jegliche Veränderungen
  - Slaves ziehen periodisch oder auf Befehl ein Abbild
- Vorteile: Redundanter (Single Point of Failure), performanter (Performance Bottleneck)
- Nachteile: Master-Copy (Single...), Gleichzeitige/Widersprüchliche Updates seitens der Slaves, Gefahr beim Ziehen der Master-Copy

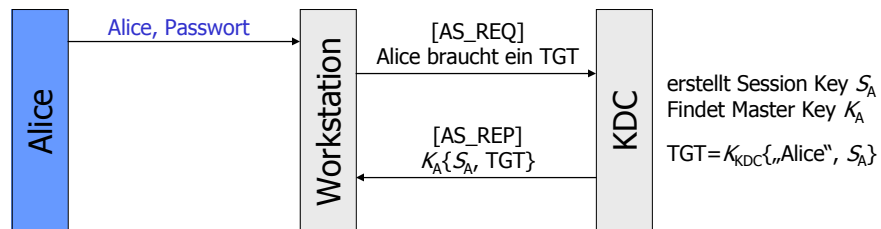
# Kerberos V4 Ablauf

## 1. Schritt: Login



# Kerberos V4 Login

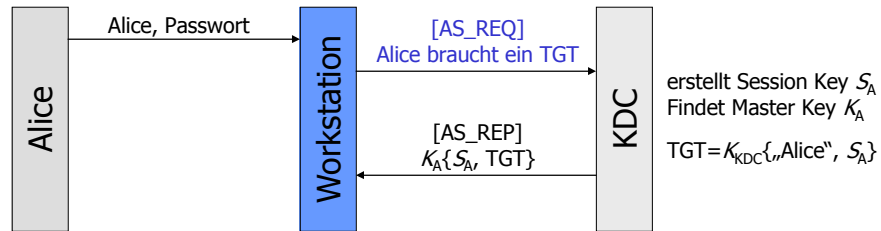
## 1. Schritt: Login (nur einmal)



- Am Workstation-Prompt meldet sich Alice mit Benutzernamen und Passwort an

# Kerberos V4 Login

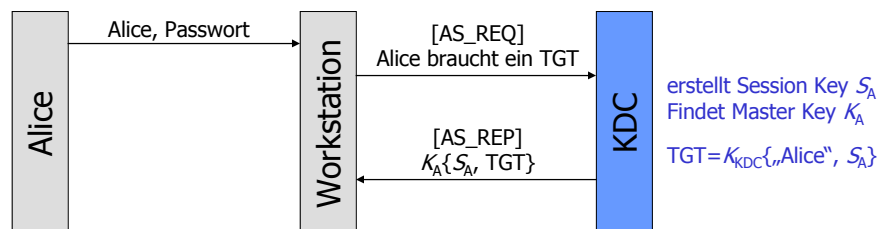
## 1. Schritt: Login



- Workstation sendet (im Klartext) nur den Benutzernamen an das KDC (AS)

# Kerberos V4 Login

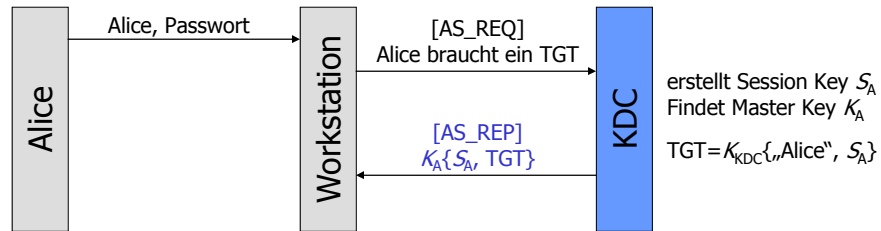
## 1. Schritt: Login



- Ticket-Granting-Ticket verschlüsselt mit KDC Master Key
  - enthält einen KDC-seitigen Session Key für die gesamte Loginsession mit Alice → ermöglicht den einmaligen Login am AS
  - enthält zusätzlich eine Gültigkeitsdauer (+IP von Alice)
  - somit nur für KDC lesbar

# Kerberos V4 Login

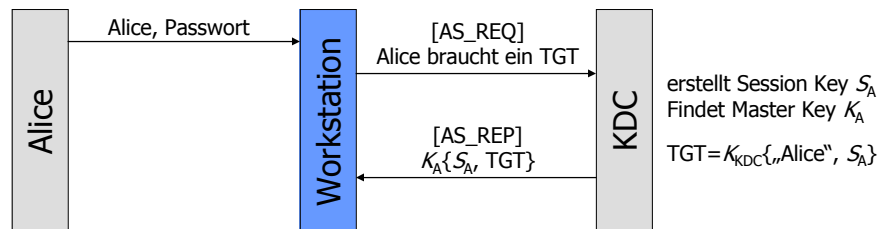
## 1. Schritt: Login



- AS\_REP mit Master Key von Alice verschlüsselt

# Kerberos V4 Login

## 1. Schritt: Login



- Workstation konvertiert das Passwort in einen DES Key und entschlüsselt damit die Credentials

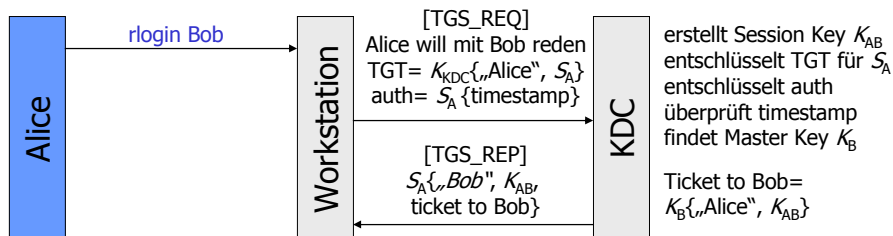
## Kerberos V4 Login

### Anmerkungen:

- In V4 wird erst das Passwort abgefragt, wenn die Credentials bereits eingetroffen sind → geringfügig sicherer (PW kürzer auf Workstation), aber auch unsicher, da man nur einen Benutzernamen an den KDC senden muss um dann das REP zu entschlüsseln
- Preauthentifikation (deaktiviert per Standard) in V5: Abfrage noch bevor AS\_REQ (jetzt jedoch mit PW verschlüsselt) gesendet wird → somit Check bevor AS\_REP zurückgesendet wird
- TGT enthält alle Daten von Alice's Loginsession und ermöglicht somit über das KDC Zugang zu allen gewünschten Remotehosts

## Kerberos V4 Remote

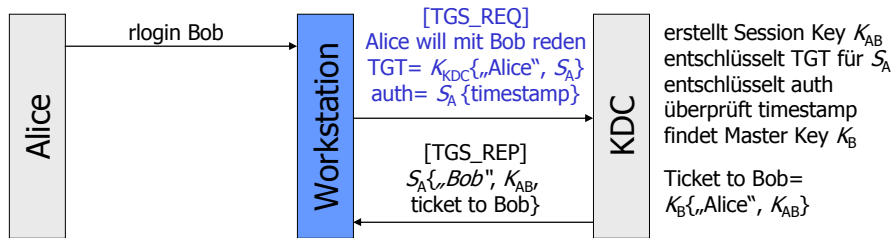
### 2. Schritt: Remote Note (für jeden Service jeweils)



- Alice (Client) benutzt eine Anwendung (hier rlogin), die Zugriff auf ein Remote Note (Server) benötigt

# Kerberos V4 Remote

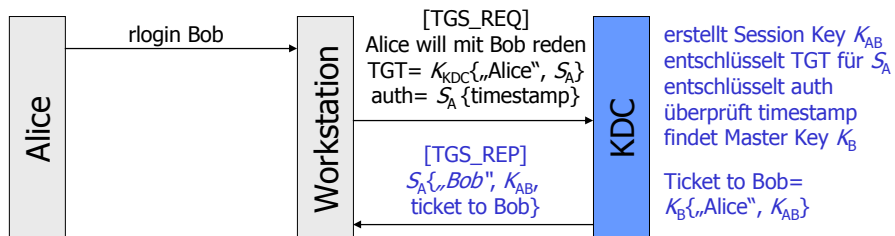
## 2. Schritt: Remote Note



- auth = Authenticator beweist gegenüber KDC, dass die Workstation den Session Key hat
- Zeitmarke setzt synchrone Uhren voraus (Abweichung def.)
  - enthaltende Zeit: Zeitpunkt der Verschlüsselung mit  $S_A$

# Kerberos V4 Remote

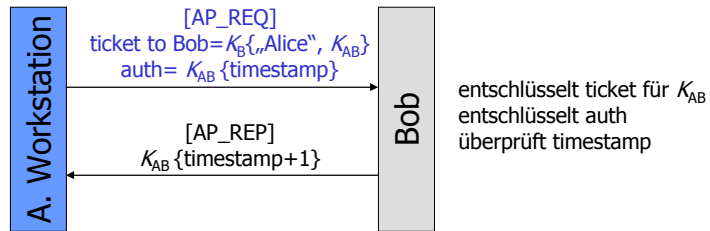
## 2. Schritt: Remote Note



- Ticket to Bob
  - enthält zusätzlich eine Gültigkeitsdauer (+IP von Alice)
- Workstation entschlüsselt mit  $S_A$  und kann nun Bob direkt ansprechen... →

# Kerberos V4 Remote

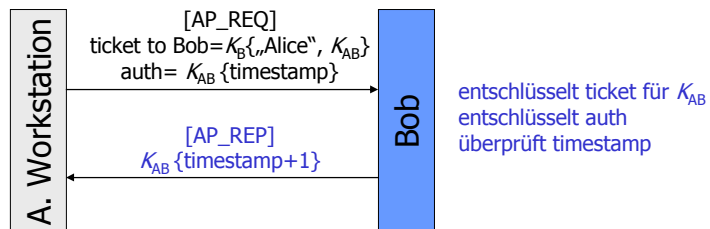
## 2. Schritt: Remote Note



- Authenticator beweist gegenüber Bob, dass die Workstation den Session Key  $K_{AB}$  hat
- enthaltene Zeit: Zeitpunkt der Verschlüsselung mit  $K_{AB}$

# Kerberos V4 Remote

## 2. Schritt: Remote Note



- Bob überprüft und erhöht die Zeitmarke um 1 als Bestätigung zur gegenseitigen Authentifikation und Kommunikation



## Kerberos V4 Remote

### Anmerkungen:

- Nach der Authentifikation ... nun in Abwägung von Geschwindigkeit und Sicherheit:
  - Unverschlüsselte,
  - Integrität schützende (Checksumme),
  - oder verschlüsselte + Integrität schützende (PCBC)
 Kommunikation zwischen Alice und Bob

→ andere Vorträge ... "Kryptografie" ...



## Message Formats

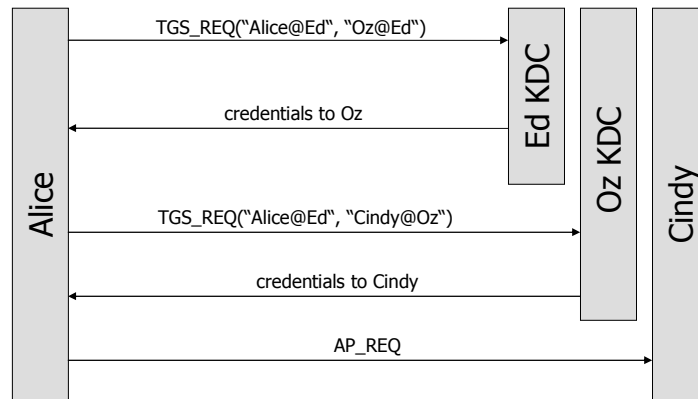
- [AS\_REQ], [AS\_REP], Tickets usw. sind alles Message Formate

Am Beispiel des Ticket Paketes:

Octets		
1		B
<= 40	Alice's name	Null-terminated
<= 40	Alice's instance	Null-terminated
<= 40	Alice's realm	Null-terminated
4	Alice's Network Layer address	
8	Session Key Alice->Bob	
1	ticket lifetime, units of five minutes	
4	KDC's timestamp, when ticket made	
<= 40	Bob's name	Null-terminated
<= 40	Bob's instance	Null-terminated
<= 7	Pad of 0's to make ticketlength mutiple to 8 octets	

## Interrealm-Authentifikation

- Principals in verschiedenen Realms authentifizieren (V4) (←)



## Kerberos V5

- Protokollablauf größtenteils wie in V4 (z.B. Gültigkeit der Tickets durch zwei Zeiten), wobei jedoch die Datenrepräsentation der ausgetauschten Nachrichten vollständig anders ist
- Nachrichtenformat in ASN.1 (Abstract Syntax Notation 1)  
→ Standardisierte Spezifikation der Datentypen  
Beispiel: `HostAddress ::= SEQUENCE{  
    addr-type[0] INTEGER  
    address[1] OCTET STRING}`
- Kerberos benutzt ASN.1 BER (Basic Encoding Rules) als Kodierungsschemas für die Wandlung in Bitsequenzen



## Kerberos V5

- Vorteile: Definition von neuen/erweiterten Datenfeldern, maschinenunabhängiges Format
- Nachteile: ASN.1 erzeugt viel Overhead, Kodieren/Dekodieren
- Neu hinzugekommene Tickettypen:  
Delegation von Rechten mit:
  - *Forwardable TGT*: ermöglicht Rechnerwechsel, um (forwarded markierte) Tickets für den neuen Rechner anzufordern
  - *Proxiabile TGT*: ähnlich Forwardable, aber wesentlich eingeschränkterEbenfalls neu (jedoch nicht für TGT's):
  - *Renewable*: Tickets mit Zusatzfeld können zum TGS zurückgesendet werden, um zeitlich aufgefrischt zu werden
  - *Postdated*: Tickets die zu bestimmten Zeiten erst gültig werden



## Systemschwächen

- Kein Schutz vor Trojaner und Co., sowie menschlichen Spionen
- Passwort schlecht → Schutz schlecht
  - Abhilfe durch zusätzliche Verfahren wie z.B. SmartCards
- Speicherung des geheimen Schlüssels des Anwendungsservers bei sich selbst (Ticket zu Bob ist mit Bob's Key verschlüsselt ↵)
- Gültigkeitsdauer der Tickets
  - Wenn zu kurz: häufig neuer Login nötig
  - Wenn zu lang: wertvolle Zeit für den Angreifer
  - TGT's sind nicht erneuerbar oder vordatierbar
- Zentralisierter Master KDC kann anfällig sein und setzt vor allem großes Vertrauen in die Administratoren voraus



## Fazit

---

- Kerberos bietet eine...
  - weitestgehend akzeptierte,
  - recht sichere,
  - gegenseitige (Client<->Server),
  - delegierbare und
  - interoperabilitäre (systemübergreifende) Authentifikationsmethode



## Quellen

---

- C. Kaufman, R. Perlman, M. Speciner; Network Security – Private Communication in a public World; 2nd Edition; Prentice Hall 2002
- M. Raeppe; Sicherheitskonzepte für das Internet; Dpunkt Verlag 1998
- W. Böhmer; VPN – Virtual Private Networks; Hanser 2002