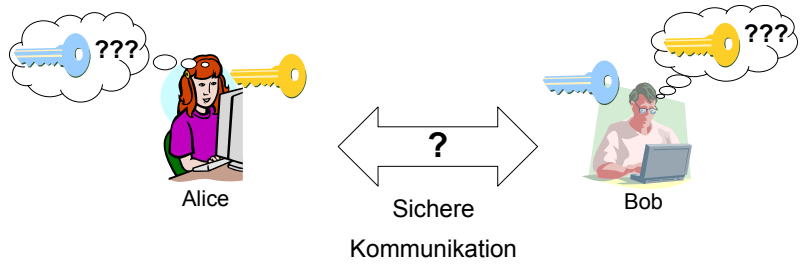
Netzsicherheit: Architekturen und Protokolle
Kerberos

1. Einführung
2. Kerberos Version 4
3. Kerberos Version 5

Schlüsselspiel



zentrale Frage: Woher kommt das Schlüsselmaterial?

•Wie kann Kommunikation zwischen Kommunikationspartnern gesichert werden?

•Anzahl der Schlüssel bei Kommunikation wenn n potentielle Kommunikationspartner miteinander reden wollen?

•Wie ist Anzahl der Schlüssel bei Public-Key-Kryptographie?

•Wie kann Kommunikation über eine dritte Partei verwaltet werden?

Einführung in Kerberos

- Kerberos: Protokoll, das den Zugriff auf Ressourcen schützt
- Ziele von Kerberos
  - **Authentifizierung**
    - Anmeldung mittels Benutzernamen und Passwort
  - **Autorisierung**
    - Zugriff auf Ressourcen durch Rechtesystem beschränkt
    - Rechte werden von einem Administrator vergeben
  - **Accounting**
    - Protokollierung von Ressourcen-Nutzung
- Single-Sign-On für Netzwerke

•Kerberos möchte folgendes Problem lösen

- •Wie kann der Zugriff auf Netz-Ressourcen gesteuert werden?
- •Wie kann ein Benutzer zum Zugriff auf Ressourcen autorisiert werden und wie funktioniert dieser Zugriff (insbesondere: Schlüsselverteilung für diesen Zugriff)?

•Kerberos geht dabei einen ähnlichen Weg wie Mehrbenutzer-Betriebssysteme, z.B. Linux.

- •Bei Linux meldet sich ein Benutzer zu Beginn einer Sitzung an und bleibt für die Dauer der Sitzung angemeldet.
- •Man spricht in diesem Fall von „Single-Sign-On“.

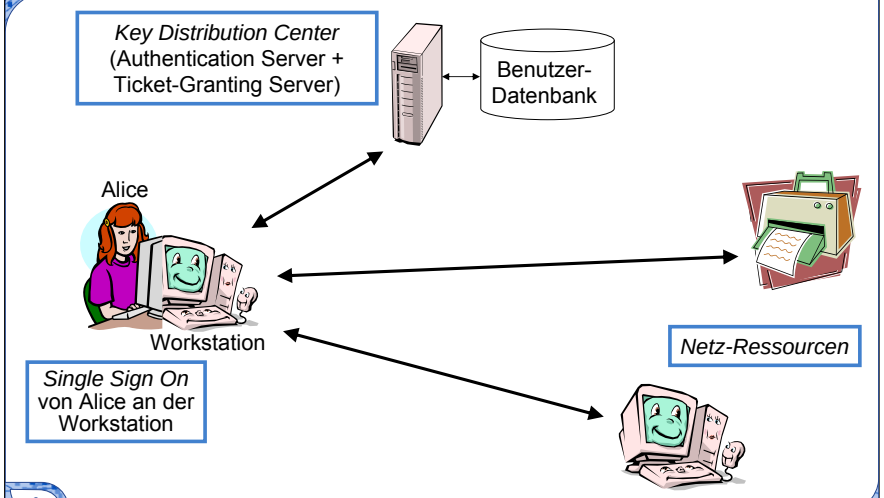
•Bei Kerberos möchte man ein „Single-Sign-On Netzwerk“ mit folgenden Eigenschaften erreichen

- •Einmaliges Authentifizieren: Einloggen mit Username und Passwort (Login Session = Zeit zwischen Ein- und Ausloggen)
- •Zugriff auf Netz-Ressourcen nur nach erfolgreicher Authentifizierung
- •Zugriffsbeschränkung durch den Administrator

•Begriffe

- •Authentifizierung: Vorgang, bei dem die Identität eines Subjekts nachgewiesen wird.
- •Autorisierung: Es sollen lediglich autorisierte Instanzen Zugriff auf bestimmte Dienste oder Daten erhalten.
- •Accounting: Sammeln von Daten über Ressourcen-Nutzung, z.B. zum Zweck der Abrechnung.

Kerberos – Überblick



•Alice: Benutzer

•Workstation: Kerberos wurde für Umgebungen entworfen, in denen sich ein Benutzer (hier: Alice) an einer Workstation mit Username und Passwort anmeldet.

•Key Distribution Center (KDC): Das KDC läuft auf einem Server. Alice Workstation verwendet Username und Passwort um Informationen (Credentials) vom KDC zu bekommen, die den Zugriff auf Netz-Ressourcen ermöglichen.

•Netz-Ressourcen: Alice möchte während einer Login Session (= Zeit zwischen Anmeldung an der Workstation und Abmeldung) Zugriff auf die Netz-Ressourcen (z.B. Drucker oder Server).

Kerberos Komponenten

- *Authentication Server* (AS)
  - Authentifizierung der Benutzer
  - Ausstellen eines Authentifizierungs-Tokens
    - *Ticket-Granting-Ticket* (TGT)
- *Ticket-Granting Server* (TGS)
  - Ressourcen-Zugangs-Server
  - Autorisierung des Ressourcen-Zugriffs bei Vorlage eines gültigen TGTs
  - Ausstellen von Zugangsberechtigungen (Tickets)
- Benutzer-Datenbank
  - Speichert Client Master Secrets aller Benutzer und Ressourcen

•Die zentralen Komponenten in Kerberos sind der Authentication Server (AS) und der Ticket-Granting Server (TGS)

•Der AS authentifiziert den Benutzer. Nach der Authentifizierung stellt der AS dem Benutzer ein Authentifzierungs-Token (TGT = Ticket Granting Ticket) aus, das der Benutzer später verwenden kann, um so genannte Tickets zu erwerben, die zum Zugriff auf Netz-Ressourcen befähigen.

•Der TGS autorisiert den Zugriff auf Ressourcen. Die Autorisierung erfolgt erst nach Vorlage eines entsprechenden TGT. Die Autorisierung erfolgt durch Zugangsberechtigungen, so genannte Tickets.

•TGS und AS müssen die gleiche Datenbank haben => es macht wenig Sinn, TGS und AS auf verschiedenen Rechnern laufen zu lassen (es sei denn zur Lastverteilung), deshalb werden TGS und AS im KDC zusammen gefasst.

•Die Benutzer Datenbank beinhaltet für alle Benutzer und alle Ressourcen des Systems jeweils ein Client Master Secret, d.h. einen symmetrischen Schlüssel. Dieser symmetrische Schlüssel ist also dem jeweiligen Benutzer sowie dem KDC (bzw. TGS und AS) bekannt.

Kerberos allgemein

- Kerberos Versionen
  - Version 1 bis 3 heute nicht mehr im Einsatz
  - Version 4 und Version 5 konzeptionell ähnlich
  - dennoch erhebliche Unterschiede: Version 4 ist
    - einfacher
    - leistungsfähiger
    - arbeitet allerdings nur in IPv4 Netzen
- Anwendungen, die Kerberos unterstützen
  - Telnet
  - BSD rtools
  - NFS
  - SSL, SSH
  - OSF/DCE
  - Windows 2000, 2003, XP, Vista

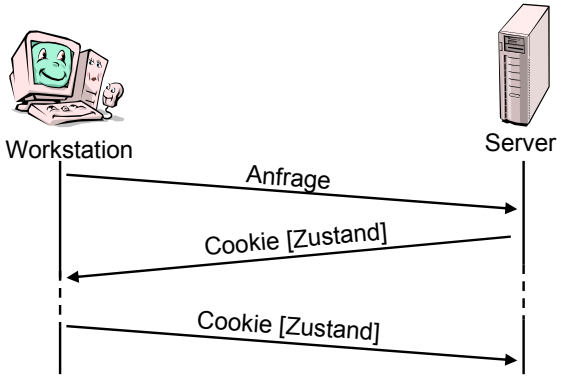
•Kerberos findet breite Anwendung, z.B. in Telnet, rtools, NFS, SSL, SSH, OSD/DCE und Windows.

•Unter Windows 2003 Server ist Kerberos der bevorzugte Authntifizierungsmechanismus für Windows Domänen

•http://www.microsoft.com/windowsserver2003/technologies/security/kerberos/default.mspx

Ausflug: Tokens/Cookies

- Server speichert Zustand pro Anfrage
  - Problem: Überlastung des Servers
  - Lösung: Token/Cookies



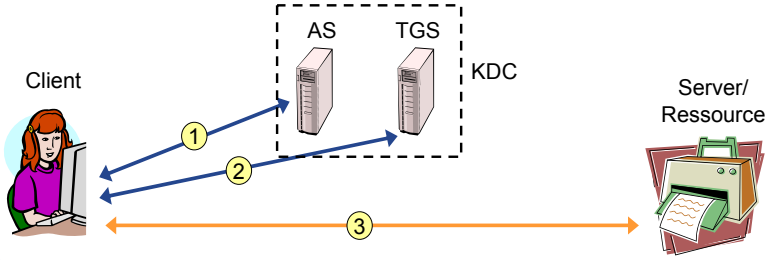
- Ein Cookie speichert einen Zustand, der sonst auf dem Server gehalten würde. Dieses Cookie muss die Workstation bei einer späteren Kommunikation wieder vorlegen und der Server kann dann daraus den Zustand übernehmen. Das Ticket Granting Ticket ist solch ein Cookie, das den Sitzungsschlüssel als Zustand enthält. So muss sich der Server diesen Schlüssel nicht merken. Wie genau ein Cookie geschützt ist wird im Folgenden an Hand des Beispiels Ticket Granting Ticket erläutert. Natürlich darf es der Workstation nicht möglich sein, den Zustand im Cookie zu ändern. Der Server muss die Cookies schützen, hierdurch wurde von der Annahme ausgegangen, dass Speicher relativ zu CPU teuer ist. Der Server muss also weniger Speicher verwenden, dafür mehr CPU.
- Zusätzlich erreicht man durch die Speicherung des Sitzungsschlüssels im TGT Robustheit gegen den Ausfall einer KDC-Instanz (siehe Replizierung des KDC).

Netzsicherheit
Architekturen und Protokolle
Kerberos

1. Einführung
2. Kerberos Version 4
3. Kerberos Version 5

Kerberos – The Big Picture

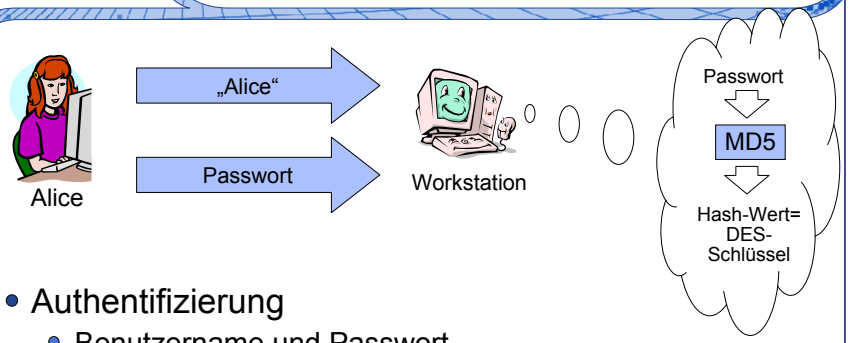
1. Anmeldung
   - Client erhält ***Ticket-Granting-Ticket*** vom ***Authentication-Server***
2. Ressourcenanforderung
   - Vorlage des Ticket-Granting-Tickets beim ***Ticket-Granting-Server***
   - Client erhält ***Ticket*** für die Ressource
3. Kommunikation mit der Ressource



•Das Ticket-Granting-Ticket ist ein Beispiel für die vorne angesprochenen Cookies: das KDC (=AS/TGS) muss keinen Zustand halten.

•Hier fällt auf, dass in Kerberos keine Kommunikation zwischen der Ressource und dem KDC notwendig ist!

Anmeldung an der Workstation



- Authentifizierung
  - Benutzername und Passwort
  - Umwandeln des Passworts in einen *DES-Schlüssel*
    - *Client-Master-Secret*
    - Umwandlung durch Hash-Funktion (MD5), siehe RFC4120

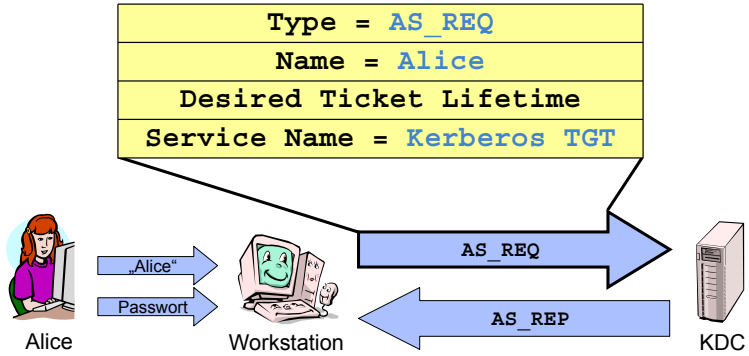
•DES ist ein symmetrischer Verschlüsselungsalgorithmus.

•MD5 ist eine kryptographische Hash-Funktion, die allerdings heute nicht mehr als sicher angesehen werden kann.

•Details zur Umwandlung des Passworts in einen DES-Schlüssel siehe RFC 4120.

Anmelden am Netz

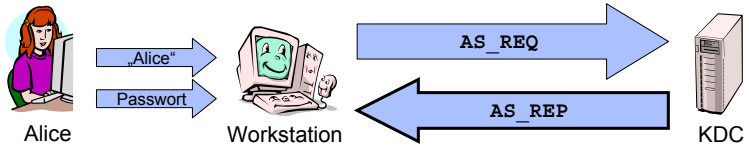
- *Authentication Server Request* (`AS_REQ`)
  - Übertragen des Benutzernamens im Klartext zum KDC



•Der Authentication Request enthält den Benutzernamen im Klartext, d.h. ein lauschender Angreifer kann diese Nachricht erkennen und kennt die Identität von Alice.

Anmelden am Netz

- Authentication Server Reply (`AS_REP`)
  - verschlüsselt mit dem Master Secret des Clients
  - *Session Key*: geheimer Sitzungsschlüssel
  - *Ticket-Granting Ticket* (TGT) enthält
    - Benutzernamen, Adresse
    - Sitzungsschlüssel, Gültigkeitsdauer
    - …



•Kerberos-Tickets sind grundsätzlich mit dem Client Master-Secret der Ressource verschlüsselt, für die das Ticket bestimmt ist.

•Auch das Ticket Granting Ticket (TGT) ist ein Ticket und ist mit dem Client Master-Secret des Key Distribution Centers (genauer: TGS) verschlüsselt ist.

Vereinfachtes Paketformat *AS_REP*

Type = *AS_REP*	Session Key (Alice↔KDC)	Ticket Granting Ticket (verschlüsselt mit Master Secret des KDC)	KDC`s Timestamp
Name = *Alice*	Name = *KDC*		
Alice`s Timestamp	Ticket Lifetime		
Ticket Expiration Time	KDC`s Key Version Number		
Credentials Length	TGT Length		

Verschlüsselt mit Master Secret des Clients

•Die beiden Timestamps sind unterschiedlich: Alice Timestamp ist der Zeitstempel aus der AS_REQ-Nachricht.

•Kritikpunkt:TGT wird doppelt verschlüsselt

•Prüfungsfrage: Wozu steht im AS_REP das Feld „KDC‘s Key Version Number“?

Ticket Granting Ticket (vereinfacht)

Client Name	`Alice`
Network Layer Address	`192.168.178.55`
Session Key	`(Alice ↔ KDC)`
Ticket Lifetime	`60`
KDC`s Timestamp	`9:20am`
Server Name	`KDC`

Verschlüsselt mit dem Master Secret des KDC

•Das Ticket Granting Ticket enthält die IP-Adresse von Alice, den Session Key, die Lebensdauer des Tickets, einen Zeitstempel sowie den Namen des KDC. Das Ticket Granting Ticket ist mit dem Client Master Secret des KDC verschlüsselt, kann also von Alice nicht gelesen werden. Das Ticket Granting Ticket dient als Cookie, damit das KDC keinen Zustand halten muss. Übermittelt Alice zu einem späteren Zeitpunkt ein TGT an das KDC, so kann das KDC daraus den Sitzungsschlüssel gewinnen und diesen für die weitere Kommunikation verwenden. Es ist also nicht notwendig, dass das KDC die Sitzungsschlüssel aller angemeldeten Benutzer speichert! Dieses Design entlastet das KDC außerordentlich!

Diskussion

- Warum tauscht Kerberos einen *Sitzungsschlüssel* aus und verwendet nicht das *Master Secret* des Client für die Kommunikation?
- Warum erfolgt der *Zugriff auf Ressourcen* über den Umweg über das TGT?

Offline-Password-Guessing Angriff

Ziel: *Erlangen des Benutzer-Passworts*

- **`AS_REQ`** und **`AS_REP`** abhören und speichern
  - eindeutig einem Benutzer zuzuordnen
  - Client Name im Klartext enthalten
- Wörterbuch-Angriff
  - pro Wort aus dem Wörterbuch
    - Wort mittels MD5 in DES-Schlüssel umwandeln
    - **`AS_REP`** entschlüsseln
    - Testen der entschlüsselten Nachricht auf Plausibilität
      - z.B. über Zeitstempel
    - Schlüsselkandidaten an weiteren Nachrichten testen
  - ist Sitzungsschlüssel bekannt, können alle weiteren Nachrichten entschlüsselt werden

•Der Angriff basiert darauf, dass oft sinnvolle Wörter als Passwort verwendet werden => Menge von Wörtern ausprobieren. Diese Menge ist wesentlich kleiner als die Menge aller möglichen Kombinationen für das Passwort => Angriff hat eine geringere Komplexität.

•Erfolgreicher Angriff hat folgende Auswirkungen

- •Inpersonifizieren des Benutzers jederzeit möglich
- •Erhalt eines TGTs
- •Nutzung von Ressourcen in dessen Namen
- •Belasten seines „Kontos“

•Hinweis für mündliche Prüfungen: Dieser Angriff sollte verstanden worden sein. Es sollte klar sein, was Kerberos 5 anders macht

Diskussion: Offline-Password-Guessing-Angriff

Ist dieser Angriff auch als *aktiver Angriff* möglich? Wenn ja, warum, wenn nein, warum nicht?

•Was ist der Vorteil des passiven Wörterbuch Angriffs gegenüber einem aktiven Angriff?

•Wie würde ein aktiver Angriff auf das System aussehen?

Kerberos Credentials

- *Ticket-Granting-Ticket*
  - ausgestellt vom Authentication-Server
  - ermöglicht Nutzung des Ticket-Granting-Servers
- *Tickets*
  - ausgestellt von Ticket-Granting-Server
  - ermöglichen Nutzung von Ressourcen
- *Authenticator*
  - Einschränkung von Replay Attacken (Angriff durch Wiedereinspielen)

•Kerberos verwendet verschiedene Arten von so genannten “Credentials” (engl. für Berechtigungsnachweis).

•Das Ticket Granting Ticket (TGT) hatten wir ja als Cookie bereits kennengelernt. Mittels eines TGTs kann ein Client beim TGS Tickets anfordern, mit denen er auf die Ressource zugreifen kann.

•Ein Ticket dient zur Nutzung einer Ressource und ist ganz ähnlich aufgebaut wie das Ticket Granting Ticket. Man kann das TGT auch als Ticket zur Nutzung der Ressource TGS ansehen.

•Weiterhin gibt es in Kerberos auch noch Authenticators, die Angriffe durch Wiedereinspielen (Replay-Attacke) verhindern sollen.

Aufbau eines Tickets

- Ticket für
  - Kommunikation von Alice mit Bob
  - angefordert von Alice

Client Name	`Alice`
Network Layer Address	`192.168.178.55`
Session Key	`(Alice ↔ Bob)`
Ticket Lifetime	`60`
KDC's Timestamp	`9:20am`
Server Name	`Bob`

Verschlüsselt mit Client Master Secret von Bob

•Alternative Schreibweise: TClient, Server = {Server, Client, Adresse, Gültigkeitsdauer, Sitzungsschlüssel KClient, Server}KServer

•Tickets sind nur vom Client (=Alice) nutzbar, der das Ticket angefordert hat. Das Ticket ist nur für die angegebene Resource (=Bob) gültig und kann während der angegebenen Lebenszeit beliebig häufig benutzt werden. Da das Ticket mit dem Master Secret des Servers (=Bobs Master Secret) verschlüsselt ist, kann es vom Client (=Alice) nicht gelesen werden!

Kerberos Credentials (2)

- Authenticators
  - erzeugt von Alice (=Client)
  - nur *einmal* einsetzbar
  - Verhinderung von Replay-Angriffen
  - Bedingung: Synchronisation der Systemuhren
    - nur in Zeitfenster gültig

Name = `Alice`
Zeitstempel

verschlüsselt mit dem jeweils verwendeten Schlüssel
(z.B. Session Key Alice↔KDC oder Session Key Alice↔Bob)

•Ziel des Authenticators ist im weiteren Einsatz, Replay-Angriffe zu verhindern. Alternative Schreibweise: AClient, Server = {Client, Zeitstempel}KClient, Server

•Synchronisation der Uhren von Alice und Bob notwendig, ansonsten können Alice und Bob nie kommunizieren

•In Kerberos v4 ist der Zeitstempel in Schritten von 5min

Adressen in Tickets

- Netzwerk-Adresse des Clients in jedem Ticket
  - Vergleich der Absender-Adresse mit der enthaltenen Adresse bei Empfang eines Tickets
    - keine Weitergabe von Tickets möglich
    - Schutz vor *Ticket-Diebstahl*
    - Verhinderung der Nutzung eines abgefangenen Tickets
- Problem
  - Fälschung der Absender-Adresse einfach
    - kein wirksamer Sicherheitsmechanismus
  - Rechteübertragung in Kerberos v4 nicht möglich
    - gewünscht, z.B. Batch-Prozess, der auf eigene Daten zugreift

•Zur Möglichkeit, IP Absender-Adressen zu fälschen (sogenanntes IP Spoofing) siehe z.B. http://spoofer.csail.mit.edu/summary.php

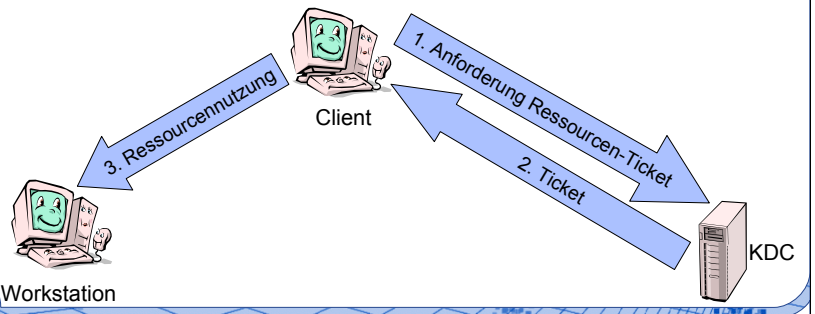
Zusammenfassung Anmeldevorgang

- KDC authentifiziert Alice anhand
  - Kenntnis des Client Master Secrets von Alice
    - aus Passwort abgeleitet
    - in Benutzer-Datenbank des KDC
- Ticket-Granting-Ticket
  - KDC kann damit vorherige Authentifizierung überprüfen
  - Auslagerung des Server-Zustands
- Alice und KDC verfügen nach Anmeldevorgang über einen Sitzungsschlüssel
  - Client Master Secret muss nicht mehr verwendet werden
  - langlebiges Geheimnis geschützt
  - Sitzungsschlüssel in TGT

•Das KDC hat eine Datenbank mit allen Client Master Secrets und Benutzernamen

Ressourcenanforderung

- Ressourcen-Nutzung nach Wiedervorlage d. TGT beim KDC
  - Ticket-Granting-Server (TGS) gibt Tickets aus
  - Zugangskontrolle durch jede Ressource
  - Erweiterung: Zugriffsbeschränkung durch KDC
    - Ausstellung eines Tickets anhand zusätzlicher Informationen



•Das KDC gibt Tickets zum Zugriff auf die Ressource aus

•Das KDC verlangt dazu die Wiedervorlage des TGT. Da in dem TGT alle Information über die vorausgegangene Authentifizierung enthalten ist, sowie der Session Key, ist es für das KDC nicht notwendig, Zustand über die Login-Session zu halten

•Das Ticket dient lediglich dazu, die Authentifizierung bei der Ressource vorzunehmen. Die Zugangskontrolle kann auf Basis von dieser Authentifizierung von jeder Ressource einzeln vorgenommen werden

•Hinweis für mündliche Prüfungen: Es sollte klar sein, welche Vorteile TGTs und Tickets bringen!

Anfordern eines Ressourcen-Tickets

→ Alice möchte ein Ticket für Bob

- *Ticket-Granting Server Request* (**`TGS_REQ`**)
  - enthält TGT
  - Ressourcen-Name
  - Authenticator verschlüsselt mit Sitzungsschlüssel
- Überprüfung durch Ticket-Granting-Server
  - Absenderadresse
  - Name
  - Zeitstempel

Type = `TGS_REQ`
KDC`s Key Version Number
TGT
Authenticator
Alice`s Timestamp
Desired Ticket
Server Name = Bob

•Da der Sitzungsschlüssel im TGT enthalten ist, muss der Ticket-Granting-Server keinen Zustand für die Sitzungsschlüssel speichern!

•Der Authenticator dient zum Schutz vor Angriffen durch Wiedereinspielen (Replay-Angriff).

•Die Überprüfung durch den Ticket Granting Server läuft folgendermaßen ab

•Entschlüsseln des TGT mit KDC Master-Secret

•Erhalten des Sitzungsschlüssel

•Überprüfung der Absenderadresse des TGS_REQ mit der TGT-Adresse

•Entschlüsseln des Authenticators mit dem Sitzungsschlüssel

•Vergleich des Namens mit dem TGT-Namen

•Prüfen des Zeitstempels

Anfordern eines Ressouren-Tickets (2)

Ticket-Granting Server Reply (**`TGS_REP`**)

- Session Key Alice ↔ Bob
- Ticket verschlüsselt mit Master Secret der Ressource

Type = TGS_REP	Session Key Alice↔Bob	Ticket für den Ressourcen-Zugriff auf Bob (verschlüsselt mit Bobs Master Secret)	KDC`s Timestamp
Name = Alice	Name = Bob		
Alices Timestamp	Ticket Lifetime		
Ticket Expiration Time	Bob`s Key Version Number		
Credentials Length	Ticket Length		

Verschlüsselt mit dem Session Key Alice ↔ KDC

•Die Antwort des KDC ist mit dem aktuellen Sitzungsschlüssel zwischen Alice und dem KDC verschlüsselt. In dieser verschlüsselten Antwort ist ein vom KDC erzeugter Sitzungsschlüssel zwischen Client (=Alice) und Ressource, auf die zugegriffen werden soll, enthalten. Außerdem ist ein Ticket in der Antwort enthalten. Dieses Ticket enthält Informationen für die Ressource und kann von Alice nicht gelesen werden, das es mit dem Client Master Secret der Ressource verschlüsselt ist.

•Der Aufbau des TGS_REP ist der gleiche wie der des AS_REP!

•Key Version Number Feld: wird verwendet wenn Password Änderung während aktiver Sitzung geschieht. Das alte Passwort wird also nicht gleich verworfen, sondern für die länge der Sitzung.

•Das Zeitstempel Feld ist 8bit, und es wird in 5 min Zeitintervallen gerechnet, also insgesamt etwas über 21h

Zusammenfassung Ressourcen-Anforderung

- Wiedervorlage
  - Ticket Granting Ticket beim Ticket Granting Server
  - Ticket Granting Server muss *keinen Zustand halten*
  - Wiedergewinnung des Sitzungsschlüssels Alice ↔ KDC
- Tickets zum Zugriff auf Ressource
  - Ausgestellt durch Ticket Granting Server
  - Enthält vom Ticket Granting Server erzeugten Sitzungsschlüssel Alice↔Bob

•Zustand ist komplett im Ticket mit eingepackt (Cookie Prinzip)

Kommunikation mit der Ressource

- *Application Request* (`AP_REQ`) enthält
  - Ticket und Authenticator
  - Überprüfung durch Ressource analog zu Überprüfung eines TGT durch TGS

Type = `AP_REQ`	Ticket	Authenticator
Bob`s key version number	(verschlüsselt mit Master Secret von Bob)	(verschlüsselt mit Session Key Alice↔Bob)

•Auch hier dient der Authenticator wieder dem Schutz vor Angriffen durch Wiedereinspielen (Replay-Angriffe)

Kommunikation mit der Ressource

- *Application Reply* (`AP_REP`)
  - enthält Authenticator
  - danach Austausch der Anwendungsdaten
    - Ungeschützt, Integritätsschutz oder Verschlüsselung mit Integritätsschutz, … → Aufgabe der Anwendungsprotokolle
- `AP_REP` eigentlich in Dokumentation nicht erwähnt, aber von vielen Anwendungen so verwendet

•Auch hier dient der Authenticator wieder dem Schutz vor Angriffen durch Wiedereinspielen (Replay-Angriffe).

•Kerberos regelt nicht, wie die Anwendung nach dem AP_REP kommuniziert! Insbesondere legt Kerberos nicht fest, ob danach ungeschützt kommuniziert wird oder ob Integrität und/oder Vertraulichkeit geschützt werden sollen. Dies ist die Aufgabe anderer Protokolle.

Diskussion: Kerberos in großen Netze

Welche Probleme können Sie sich vorstellen, wenn Kerberos in einem großen Netz eingesetzt wird?

Schlüssel-Server für große Netze

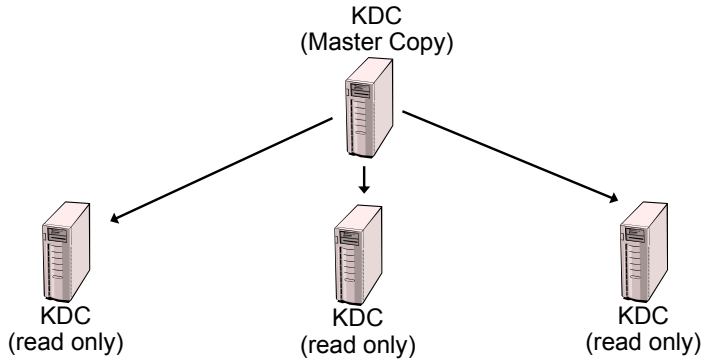
- Einzelner KDC ist *Single-Point-of-Failure*
  - Replizierung des Schlüssel-Servers
- Zentraler Punkt: Wissen alles Master-Secrets
  - Gliederung des Netzes in Domänen
    → so genannte *Realms*

•Problem 1: Ein KDC als Single-Point-of-Failure bedeutet, dass bei Ausfall des KDC bzw. der Verbindung dorthin keine Anmeldungen mehr möglich sind. Für bereits angemeldete Workstations ist es nicht möglich, neue Tickets für Ressourcen zu erhalten, also ist für diese Clients kein Zugriff auf die Ressourcen mehr möglich. Bereits vorhandene Tickets können aber nach wie vor für den Zugriff verwendet werden. Darüber hinaus stellt ein einzelnes KDC einen Flaschenhals dar, d.h. eine Überlastung dieses Systems hat Auswirkungen auf das gesamte Netz.

•Problem 2: Wenn ein jedes KDC über alle Client Master Secrets verfügt, setzt dies voraus, dass alle volles Vertrauen haben in den Betreiber der KDCs. Darüber hinaus enthüllt ein Einbruch in eines der KDCs die gesamten Client Master Secrets. Weiterhin entsteht auch ein Vertrauensproblem, da jeder dem Server vertrauen muss, von wem wird dieser bei einem großen Netz betrieben? Kann es in einem großen Netz Instanzen geben, welchem alle vertrauen? (denke an Abteilungen oder Fakultäten …)

Replizierte Schlüssel-Server

- Alle KDCs besitzen gleiches KDC Master-Secret
  - ein Master Copy der Benutzerdatenbank
  - ein oder mehrere *read-only*-Slaves



•Hauptproblem ist, wie die Benutzerdatenbank repliziert werden kann

•In der Benutzerdatenbank ist zu einem Benutzernamen das entsprechende Client Master Secret gespeichert, also z.B. ist unter dem Benutzernamen “Alice” das Master Secret von Alice abgespeichert, das wie vorne besprochen aus dem Passwort hergeleitet wird.

Replizierte Schlüssel-Server

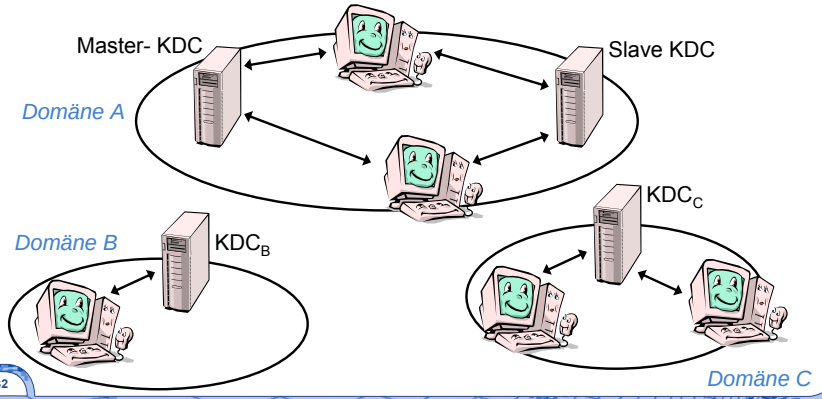
- Master-Copy der Benutzerdatenbank
  - alle Änderungen auf der Master-Copy
  - Authentifizierung über Master-Copy KDC und read-only KDC
  - Ausfall des Masters
    - keine Update-Operationen möglich
    - Netz bei Ausfall des Masters aber weiter nutzbar
- Synchronisierung der read-only Slaves
  - periodisch oder per Administrations-Kommando
  - „Klartext-Übertragung“ mit anschließendem kryptographischen Hash
    - Client Master Secrets verschlüsselt mit KDC Master-Secret
    - Hash zum Schutz vor Manipulation, Vertauschen, Anfügen von Daten

•Bei Übertragung der Daten muss Vertraulichkeit und Integrität geschützt werden

•Vertraulichkeit alleine reicht eventuell nicht. Werden z.B. die Werte einzeln übertragen, z.B. „Alice“, Master Secret Alice (verschlüsselt), „Bob“, Master Secret Bob (verschlüsselt), dann könnte ein Angreifer z.B. die Master Secrets der beiden vertauschen. Ist „Alice“ der Angreifer, so kann sie nach dem Angriff mit ihrem eigenen Passwort Bob‘s Identität übernehmen (wegen der Vertauschung!)

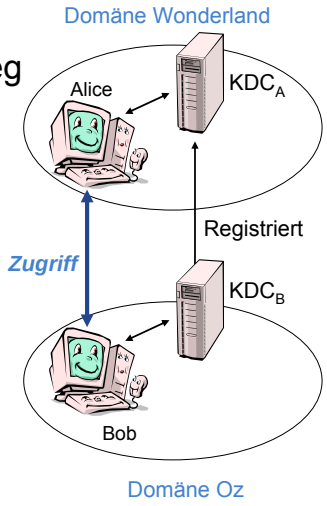
Domänen

- Lösung für viele administrative Bereiche: *Domänen*
  - eigene Benutzer-Datenbank für jede Domäne (Realm)
  - innerhalb der Domäne Replizierung möglich
  - KDCs einer Domäne besitzen gleiches KDC Master-Secret



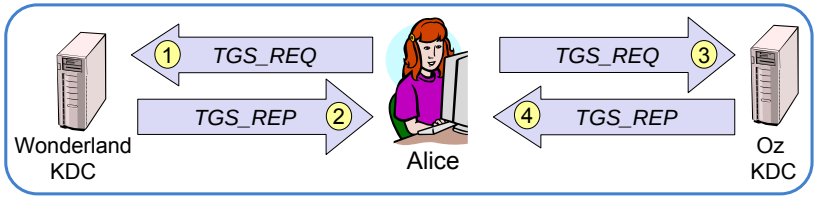
Inter-Domänen-Authentifizierung

- **Problem:** Authentifizierung über Domänengrenzen hinweg
  - Nutzung von Ressourcen in anderer Domäne
  - Autorisierung durch KDC der anderen Domäne
- **Lösung:** KDC kann als Client eines anderen KDC registriert sein



Inter-Domänen-Authentifizierung (2)

- Alice@Wonderland möchte mit Bob@Oz kommunizieren
  1. Alice fordert Ticket für KDC der Domäne Oz an
  2. Wonderland-KDC erstellt Ticket für Oz-KDC
  3. Alice fordert Ticket für Bob von Oz-KDC an
     - Ticket von Wonderland-KDC als TGT
  4. Oz-KDC erstellt Ticket, mit dem Alice auf Bob zugreift



- Verkettung von Inter-Domänen Tickets in Kerberos v4 nicht möglich
  - Domänen-Feld des Ticket und Absender-Domäne müssen übereinstimmen

•Das Domänen-Feld wurde in den vorherigen Abbildungen weggelassen

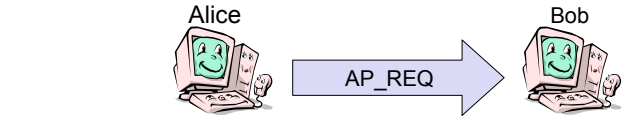
•Hinweis für mündliche Prüfungen: Dieser Vorgang sollte an einem Beispiel gezeigt werden können

•Das Konzept ist keine hierarchische Authentifizierung, die KDCs werden müssen untereinander mit dem normalen Kerberos verfahren registriert sein

•Auf einem KDC muss explizit konfiguriert werden wer auf der fremden Domäne welche Rechte besitzt

Passwort-Änderung

- Passwort-Änderung
  - Benutzer eines OS kann jederzeit Passwort ändern
    → Änderung beeinflusst nur ihn
  - Änderung des Client Master-Secrets genauso einfach?
  - **Problem**: ausgestellte Tickets sind mit Master-Secret verschlüsselt, das aus dem alten Passwort generiert wurde!



Ticket für Bob ist mit dessen Master-Secret verschlüsselt

Änderung des Passworts → neues Master-Secret

- **Frage**: werden alle bereits ausgestellten Tickets ungültig?

Passwort-Änderung (2)

- Lösung: *Versionsnummer der Schlüssel*
  - Speicherung mehrerer Schlüsselversionen
    - Gültigkeitsdauer eines Tickets auf 21,25 Stunden beschränkt
  - Jedes Ticket, jede Nachricht enthält Versionsnummer
    - ID des verwendeten Schlüssels
- Problem: *Replizierung des KDCs*
  - Verteilung des neuen Master-Secrets auf Slave-KDCs
  - einloggen mit neuem Passwort unter Umständen nicht sofort möglich
  - altes Passwort weiterhin gültig

  → Verwirrung des Benutzers

•Die Versionsnummer eines verwendeten Schlüssels steht jeweils in den Feldern “Key Version Number“, siehe vorne.

•Die Gültigkeit des Tickets ist auf 21,25 Stunden begrenzt, weil im Ticket ein Feld fester Länge für die Gültigkeit (in Einheiten von 5 Minuten!) vorgesehen ist und 21,25 Stunden ist für dieses Feld der maximal mögliche Wert.

Zusammenfassung Kerberos v4

- Single-Sign-On-Network
  - Authentifizierung mit Benutzernamen und Passwort
  - Anmeldung beim Authentication-Server
  - Anforderung der Ressourcenutzung beim Ticket-Granting-Server
  - Autorisierung durch den Ticket-Granting-Server
  - Schutzmechanismen der Anwendung nicht festgelegt

•Kerberos kümmert sich ausschließlich um die gegenseitige Authentifizierung

•Was danach für eine Kommunikation durchgeführt wird ist nicht Teil von Kerberos

Zusammenfassung Kerberos v4 (2)

- KDC als Single-Point-of-Failure
  - Replizierung des KDC
    - 1x Master-Copy User-Datenbank, beliebige Read-Only-Slaves
    - Problem: Update des Passworts
  - Domänen
    - Interdomänen-Authentifizierung
    - keine Verkettung
- Kerberos Netzwerktrace
  - siehe Vorlesungsmaterialien *ns-xx-KerberosTrace*

Verwenden Sie zur Prüfungsvorbereitung bitte unbedingt die Folien mit Anmerkungen!

•Die wichtigsten Take-away Points zu Kerberos

- •Zentrale Vertrauensinstanz
- •Authentifikation des Clients (Single-sign-on)
- •Server muss sich keinerlei Zustand speichern auf Grund der Cookies → kein Denial-of-Service gegen Kerberos möglich
- •Session Keys
- •Replizierung
- •Domänen

Literatur

Sichere Netzwerkkommunikation,
Bless et al., Springer, 2005.

Network Security – Private Communication in a Public World,
Kaufmann, Perlman, Speciner, Prentice Hall, 2002.

Telnet Authentication: Kerberos Version 4,
D. Borman, RFC 1411, 1993