JAVA-Simulation des Kerberos-Protokolls

In dieser Aufgabe soll eine JAVA-Simulation des Kerberos-Protokolls, das in der Vorlesung vorgestellt wurde, um die Client-Komponente und einen Teil der Applikationsserver-Komponente ergänzt werden (→ beigefügter Vorlage-Code).

Für die Simulation werden folgende vereinfachende Annahmen getroffen:

- Ein Client, ein Server, ein KDC
- Keine Realms
- Geheime Schlüssel sind zufällig erzeugte Integer-Zahlen (long)
  Zur Verschlüsselung gibt es für jedes Datenobjekt Methoden
  `encrypt (long key)` und **`decrypt(long key`**), die das Objekt anhand des angegebenen Schlüssels vollständig ver- bzw. entschlüsseln.
- Kommunikation über lokale JAVA-Methodenaufrufe
- Leichte Protokollmodifikation:
  - Im 5. Schritt wird dem Server zusätzlich ein Kommando mitgeteilt, welches dieser ausführen soll.

Um das Simulationsmodell einfach zu halten, werden folgende Objekte durch einfache JAVA-Klassen mit den angegebenen Methoden repräsentiert:

- *Simulationsumgebung aufbauen:*
  `Class KerberosSim`
  - *Simulation einer Benutzer-Session:*
    Login mit Passwort-Abfrage und Zugriff auf Fileserver:
    `public static void main (String args[]) {}`
  - *KDC, Server und Client initialisieren, d.h. geheime Client- und Serverschlüssel erzeugen und dem KDC mitteilen:*
    `public Client initKerberos() { }`
  - *Passwortabfrage vom Benutzer:*
    `public char[] readPasswd(String userName) { }`

- *Client-Funktionalität:*
  `Class Client`
  - *TGS-Ticket für den übergebenen Benutzer vom KDC (AS) holen:*
    `public boolean login(String userName, char[] password)`
  - *Bei angegegebenem Server authentifizieren und Kommando "showFile" ausführen lassen:*
    `public boolean showFile(Server fileServer, String filePath)`

- *KDC-Funktionalität (AS+TGS):*
  `Class KDC`
  - *Initialisierungsmethoden (Anlegen der Accounts vor dem ersten Anmelden):*
    - **`public long serverRegistration(String sName)`**
    - **`public void userRegistration(String userName,`**
      `char[] password)`

- *AS-Funktionalität (Anforderung eines TGS-Tickets bearbeiten):*
  - **`public TicketResponse requestTGSTicket (String userName, String tgsServerName, long nonce)`**

- *TGS-Funktionalität (Anforderung eines Server-Tickets bearbeiten):*
  - `public TicketResponse requestServerTicket (Ticket tgsTicket, Auth tgsAuth, String serverName, long nonce)`

- *(Applikations-)Server-Funktionalität:*
  `Class Server`
  - *Anmeldung des Servers beim KDC:*
    `public void setupService(KDC kdc)`
  - *Anforderung eines Service-Requests bearbeiten:*
    `public boolean requestService(Ticket srvTicket, Auth srvAuth, String command, String parameter)`

- **`Class Ticket`** *// Datenstruktur mit Zugriffsfunktionen*
- **`Class Auth(entication)`** *// Datenstruktur mit Zugriffsfunktionen*
- **`Class TicketResponse`** *// Datenstruktur mit Zugriffsfunktionen*

Das Protokoll und die Datenstrukturen (in Kurznotation) sind wie folgt,
wobei { .. } eine Verschlüsselung der Daten in { } mit geheimem Schlüssel K bedeutet.
K

	Von	An	Nachrichten-Name	Nachrichten-Inhalt
1.	Client C	KDC (AS)	**`requestTGSTicket`**	C, TGS, n_1
2.	KDC (AS)	Client C	**`TicketResponse`**	$\{K_{C\text{-}TGS}, n_1, \{C, TGS, t_1, t_2, K_{C\text{-}TGS}\}\}_{K_{TGS}\ K_C}$ (TicketResponse)
3.	Client C	KDC (TGS)	**`requestServerTicket`**	$\{C, TGS, t_1, t_2, K_{C\text{-}TGS}\}_{K_{TGS}}$ (Ticket), $\{C, t\}_{K_{C\text{-}TGS}}$ (Authentication), S, n_2
4.	KDC (TGS)	Client C	**`TicketResponse`**	$\{K_{C\text{-}S}, n_2, \{C, S, t_1, t_2, K_{C\text{-}S}\}\}_{K_S\ K_{C\text{-}TGS}}$ (TicketResponse)
5.	Client C	Applik.-Server S	**`requestService`**	$\{C, S, t_1, t_2, K_{C\text{-}S}\}_{K_S}$, $\{C, t\}_{K_{C\text{-}S}}$, Command

INF-WPP	Praktikum IT-Sicherheit	Hübner/Behnke
WS 15	Aufgabe 4 – Kerberos	20.01.2016

Aufgaben:

1. Ergänzen Sie den beigefügten Code für die folgenden Methoden der Klasse `Client`:

- **`public boolean login(String userName, char[] password) {…}`**
  Aufgabe: TGS-Ticket für den übergebenen Benutzer vom KDC (AS) holen (TGS-Servername: **`myTGS`**) und zusammen mit dem TGS-Sessionkey und dem UserNamen speichern.
  Rückgabe: Status (Login ok / fehlgeschlagen)

- **`public boolean showFile(Server fileServer, String filePath) {..}`**
  Aufgabe: Serverticket vom KDC (TGS) holen und „**`showFile`**“-Service beim übergebenen Fileserver anfordern.
  Rückgabe: Status (Befehlsausführung ok / fehlgeschlagen)

Führen Sie dabei in Ihrem Code alle Prüfungen durch, die aufgrund der dem Client zur Verfügung stehenden Informationen möglich sind!

2. Ergänzen Sie den beigefügten Code für die folgende Methode der Klasse `KDC`:

- **`public TicketResponse requestServerTicket (Ticket tgsTicket, Auth tgsAuth, String serverName, long nonce)`**
  Aufgabe: Anforderung eines Server-Tickets bearbeiten.
  Rückgabe: TicketResponse für die Anfrage

Führen Sie dabei in Ihrem Code alle Prüfungen durch, die aufgrund der dem KDC (TGS) zur Verfügung stehenden Informationen möglich sind!

3. Ergänzen Sie den beigefügten Code für die folgende Methode der Klasse `Server`:

- **`public boolean requestService(Ticket srvTicket, Auth srvAuth, String command, String parameter)`**
  Rückgabe: Status (Befehlsausführung ok / fehlgeschlagen)
  Aufgabe: „**`showFile`**“-Befehl mit Filepath als Parameter ausführen, d.h. Dateiinhalt zeilenweise auf der Konsole ausgeben.

Führen Sie dabei in Ihrem Code alle Prüfungen durch, die aufgrund der dem Server zur Verfügung stehenden Informationen möglich sind!

4. Testen Sie Ihren Code im Rahmen der Kerberos-Simulationsumgebung!
Bauen Sie geeignete Ausgabeanweisungen in Ihren Code ein

Tipps:

- *Verwenden Sie die gegebenen „print“-Methoden für Ticket, Auth und TicketResponse!*
- *Verwenden Sie die gegebenen Hilfsmethoden, insbesondere zum Ver- und Entschlüsseln!*
- *Lesen Sie die Aufgabenstellung genau durch!*

Viel Spaß!