1ère master ingénieur industriel finalité informatique
Cours de Systèmes d'exploitation (Programmation réseaux)

(2014-2015)

T.P. n°5 : Une architecture client-serveur Java classique avec sécurité logicielle hybride symétrique et asymétrique

Objectifs :

- Etre capable de déployer une architecture logicielle de type client-serveur en intégrant diverses technologies
- Savoir établir une méthodologie de développement et de test
- Savoir coordonner les actions d'une équipe
- Etre capable d'exposer (avec Powerpoint) les grands principes de l'architecture considérée en vulgarisant si nécessaire

Travail à réaliser par équipe de trois étudiants pour :

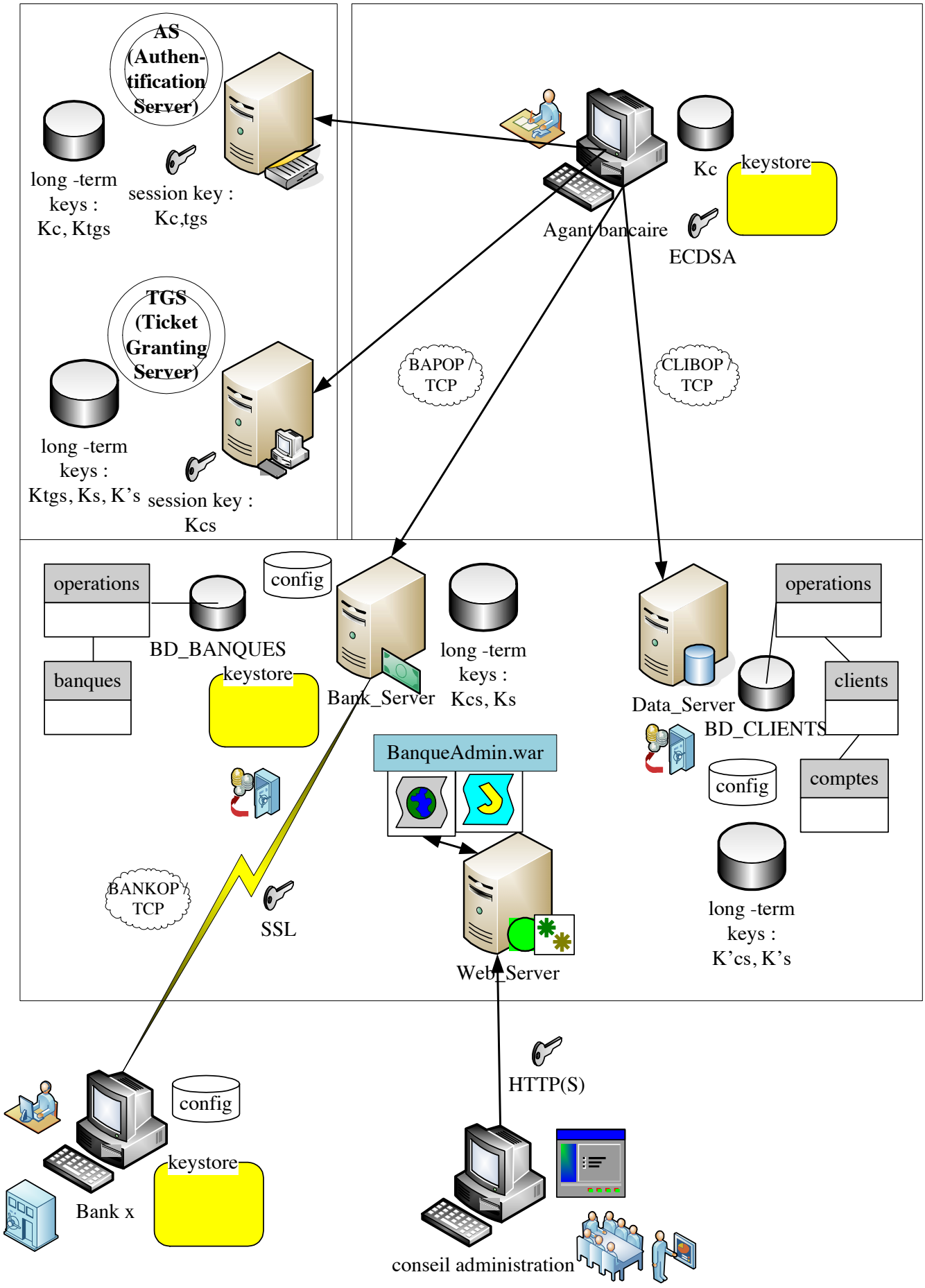
20/4/2015 : exposé de vulgarisation avec PowerPoint
27/4/2015 : le reste avec un document réduit expliquant simplement la structure du travail et les degrés de responsabilité de chaque membre du team pour chaque composant développé.

1. Etude de ThreadPoolExecutor

Il s'agit d'étudier méthodiquement la classe ThreadPoolExecutor du package java.util.concurrent qui fournit le mécanisme d'un pool de threads classiques. Il fau donc envisager les différents cas possibles en jouant sur les différents paramètres du constructeur et en envisageant aussi les différents cas d'un serveur peu utilisés, très souvent utilisé, avec un nombre d'utilisateurs plus ou moins constant ou au contraire très variable.

2. Mise en œuvre d'une architecture hybride Kerberos-ECDSA-SSL

Il s'agit ici de développer une architecture clients-serveurs dans le contexte d'une institution bancaire, selon le schéma suivant :


AS (Authen-tification Server)
long -term keys : Kc, Ktgs
session key : Kc,tgs
TGS (Ticket Granting Server)
long -term keys : Ktgs, Ks, K's
session key : Kcs
Agant bancaire
Kc
keystore
ECDSA
BAPOP / TCP
CLIBOP / TCP
operations
banques
BD_BANQUES
keystore
config
Bank_Server
long -term keys : Kcs, Ks
operations
clients
Data_Server
BD_CLIENTS
config
comptes
long -term keys : K'cs, K's
BanqueAdmin.war
Web_Server
BANKOP / TCP
SSL
HTTP(S)
config
keystore
Bank x
conseil administration

On peut noter les points suivants.

2.1) Trois serveurs sont envisagés

Thibault ♦ **Bank_Server** qui gère les opérations de crédit et de débit de la banque avec les autres banques (qui en sont donc les clients); ces intervenants ont des communications privilégiées qui sont sécurisées au moyen de SSL; les seules commandes du protocole BANKOP (BANL Operations Protocol) sont un crédit et une demande de débit; il utilise une base de données BD_BANQUES; quelques agents de la banque contrôlent et valident ces opérations ou pas, selon le protocole BAPOP - BAnk Partners Operation Protocol)

Kaoutare ♦ **Data_Server** qui gère toutes les informations sur les clients de la banque, leurs comptes, les opérations qu'ils effectuent; les clients de ce serveur sont des agents de la banque qui gèrent les opérations, les contrôlent, les valident ou pas, etc; il répond à des requêtes (protocole CLIBOP - CLIent Bank Operation Porotocol) comme :

- liste des opérations d'un client donné pour une période de temps donné;
- valeur moyenne mensuelle des comptes d'un client donné;
- nombre de débits refusés par an sur les comptes d'un client donné pour cause de solde insuffisant;
- validation d'un compte comme "client fiable" ou "client douteux" d'un client donné;

Nassim ♦ **Web_Server** (en pratique, un serveur Apache avec un moteur à servlet Apache, ou plus simplement encore un serveur Tomcat) qui fournit des informations descriptives des activités de la banque (essentiellement des chiffres et des graphiques statistiques divers); les clients sont les membres du conseil d'administration de la banque, qui sont rarement présents au siège mais plutôt un peu partout de par le monde; les requêtes sont du type :

* valeur moyenne du montant d'un débit pour un client personne physique
* valeur moyenne du montant d'un débit pour un client banque
* graphique d'évolution du delta crédits/débits pour les différentes banques partenaires
* graphique de répartition des crédits des banques partenaires ce jour
* étude de la régression-corrélation descriptive des débits en fonction du nombre de comptes (certains client en ont jusqu'à 8);

il utilise une base de données BD_CLIENTS;

2.2) L'accès des agents de banque aux différents serveurs se fait dans une architecture Kerberos à 4 acteurs (donc avec un AS et un TGS).

2.3) Quand il s'agit de valider un compte pour l'année, le comptable client de Data_Server est tenu de signer sa décision au moyen d'une signature ECDSA.

2.4) Les requêtes posées par les membres du conseil d'administration de la banque sont traitées par une application Web BanqueAdmin.war structurée en modèle MVC (et même MVC2). Les graphiques statistiques sont construits avec la bibliothèque JFreechart.

2.5) Les deux serveurs Bank et Data utilisent sont construits au moyen de ThreadPoolExecutor

2.6) Les protocoles applicatifs sont réduits à un nombre réduit de commandes afin de ne pas alourdir le travail demandé, l'intérêt majeur du dossier étant les architectures de sécurité et pas les machines à états des protocoles.

3. Questions additionnelles

3.1) Est-il envisageable de stocker les clés ECDSA dans un keystore classique ?

3.2) Utiliser pour ECDSA les clés secp192k1 et sect163k1. Expliquer les caractéristiques mathématiques de ces courbes. Utiliser pour cela, par exemple, http://www.secg.org./sec2-v2.pdf.

3.3) Comment aménager la composante Web de manière à basculer de http en HTTPS dès qu'une demande de débit ou de crédit est lancée ?

4. Méthodologie des tests réalisés

Un ou plusieurs tableaux, avec par exemple :
cas de figure envisagé - data en input - résultats en output - indications sniffer - évaluation des performances

Il n'est pas nécessaire d'avoir des dizaine es de tests : ce qui import ici est la manière de les appréhender.

5. Exposé de vulgarisation

Il s'agit de réaliser un PowerPoint expliquant au grand public, par définition non averti, les grandes étapes du paiement électronique selon le **protocole 3-D secure.**

6. A la recherche d'une méthodologie de travail en équipe

Il s'agira donc ici de structurer et de coordonner un travail sans tomber dans les travers d'un découpage horizontal sans intérêt (c'est-à-dire 3 TPs individuels mis bout à bout).

Quelques suggestions :

1) analyse globale préalable des applications finales et grandes articulations du projet;
2) principaux points à développer;
3) principaux tests locaux à prévoir;
4) définition de points de synchronisation entre les différentes tâches;
5) journal de bord : décision prises, comment, qui est responsable de quoi, quels bugs rencontrés et quelles solutions, quels tests avec succès, pourquoi tel ou tel retard.