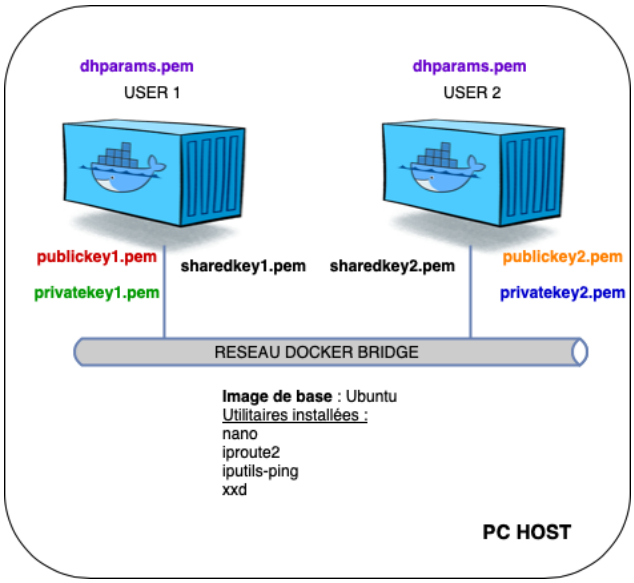
GÉNÉRATION D'UNE CLÉ PARTAGÉE AVEC DIFFIE-HELLMAN



Objectif : dans ce TP, nous allons générer une clé partagée sur 2 machines distinctes avec le protocole Diffie-Hellman en utilisant openssl.
La manipulation sera réalisée à l'aide de 2 conteneurs Docker Ubuntu sur lesquels sont installés les utilitaires nano, iproute2 et iputils-ping. L'image Docker sera réalisé à partir d'un fichier Dockerfile fourni.

1. Téléchargement du fichier Dockerfile :

Depuis un terminal Ubuntu, saisissez :

```
git clone https://github.com/bouhenic/DiffieHellman
cd DiffieHellman
```

2. Création de l'image Docker :

```
docker build –t tpdiffhell .
```

3. Création des deux conteneurs :

```
docker run –d ––name user1 –hostname user1 tpdiffhell
docker run –d ––name user1 –hostname user1 tpdiffhell
```

4. Interaction avec les conteneurs:

Intégrez le conteneur user1 :

```
docker exec –it user1 /bin/bash
```

Intégrez le conteneur user2 dans un autre onglet :

```
docker exec –it user1 /bin/bash
```

5. Génération des paramètres de Diffie-Hellman :

La première étape consiste à générer les paramètres Diffie-Hellman (un nombre premier pp et un générateur gg). Ces paramètres seront utilisés par les deux parties pour générer leurs clés. Vous pouvez générer un fichier de paramètres DH avec OpenSSL comme suit :

```
openssl dhparam –out dhparams.pem 2048
```

6. Copie des paramètres sur le conteneur user2 :

Pour cela on va d'abord copier le fichier du conteneur user1 vers la machine host. Pour cela on ouvre un autre onglet du terminal :

```
docker cp user1:/dhparams.pem .
```

Dans un second temps, on recopie ce fichier dans le conteneur user2 :

```
docker cp dhparams.pem user2:/
```

7. Génération des clés privées et publiques :

Dans le conteneur user1 (on génère ici privatekey1, puis publickey1):

```
openssl genpkey –paramfile dhparams.pem –out privatekey1.pem
openssl pkey –in privatekey1.pem –pubout –out publickey1.pem
```

Dans le conteneur user2 (on génère ici privatekey2, puis publickey2):

```
openssl genpkey –paramfile dhparams.pem –out privatekey2.pem
openssl pkey –in privatekey2.pem –pubout –out publickey2.pem
```

8. Échange des clés publiques :

Dans l'onglet du terminal de la machine host :

Copie de publickey1.pem vers le user2 :

```
docker cp user1:/publickey1.pem /
docker cp publickey1.pem user2:/
```

Copie de publickey2.pem vers le user1 :

```
docker cp user2:/publickey2.pem /
docker cp publickey2.pem user1:/
```

9. Génération de la clé partagée :

Dans le conteneur user1 (on génère ici sharedkey1.bin):

```
openssl pkeyutl –derive –inkey privatekey1.pem –peerkey
publickey2.pem –out sharedkey1.bin
```

Dans le conteneur user2 (on génère ici sharedkey2.bin):

```
openssl pkeyutl –derive –inkey privatekey2.pem –peerkey
publickey1.pem –out sharedkey2.bin
```

On peut visualiser les fichiers sharedkey1.bin et sharedkey2.bin avec l'outil de représentation hexadécimal xxd, on vérifiera visuellement qu'ils sont identiques.

```
xxd sharedkey1.bin
```

10. Chiffrement d'un message depuis user1.

Générer un fichier texte message.txt :

```
nano message.txt
```

Saisissez le texte de votre choix.

Convertissez la clé partagée en mode hexadécimal :

```
hexkey=$(xxd –p –c 32 sharedkey1.bin | tr –d '\n')
```

Chiffrer le fichier message.txt avec la clé hexkey (clé sharedkey1.bin formatée) :

```
openssl enc –aes–256–cbc –salt –in message.txt –out message.enc –
pass pass:"$hexkey" –pbkdf2 –iter 10000
```

11. Copie du message chiffré (message.enc) vers le user2.

Dans l'onglet du terminal de la machine host :

Copie de message.enc vers le user2 :

```
docker cp user1:/message.enc .
docker cp message.enc user2:/
```

12. Déchiffrage du message sur le user2 :

Dans le conteneur user2, convertir la clé partagée au format hexadécimal :

```
hexkey=$(xxd –p –c 32 sharedkey2.bin | tr –d '\n')
```

Déchiffrer le message :

```
openssl enc –aes–256–cbc –d –in message.enc –out message–
dechiffre.txt –pass pass:"$hexkey" –pbkdf2 –iter 10000
```

Vérifier le contenu du message déchiffré :

```
cat message-dechiffre.txt
```