

OBTENIR



# Les mobiles au sein d'un réseau de laboratoire



Serge Bordères

Centre d'Etudes Nucléaires de Bordeaux-Gradignan  
Observatoire des TEchnologies Nomades et de l'Internet pour la Recherche

ANF Mathrice - Marseille - 10 octobre 2016

# Sommaire

---

I - Le contexte des mobiles

II- Architecture des systèmes  
d'exploitation des mobiles

III - Fonctionnalités

IV - Sécurisation d'un service  
avec un smartphone (SSH/VPN)



---

## Le contexte des mobiles

---

# Le contexte de mobiles

---

- La principale caractéristique aujourd'hui du monde des mobiles c'est l'usage de matériels personnels en relation avec l'activité professionnelle.
- Qui par extension encourage l'usage d'ordinateurs « classiques » personnels.
- Apparition d'un nouvel Acronyme : **B.Y.O.D**
- Normalement, le BYOD est une politique volontaire (et parfois incitative) d'un établissement pour encadrer l'usage de matériels personnels
- Dans notre milieu, c'est plutôt une décision de l'utilisateur...mais par extension on appelle ça du BYOD.

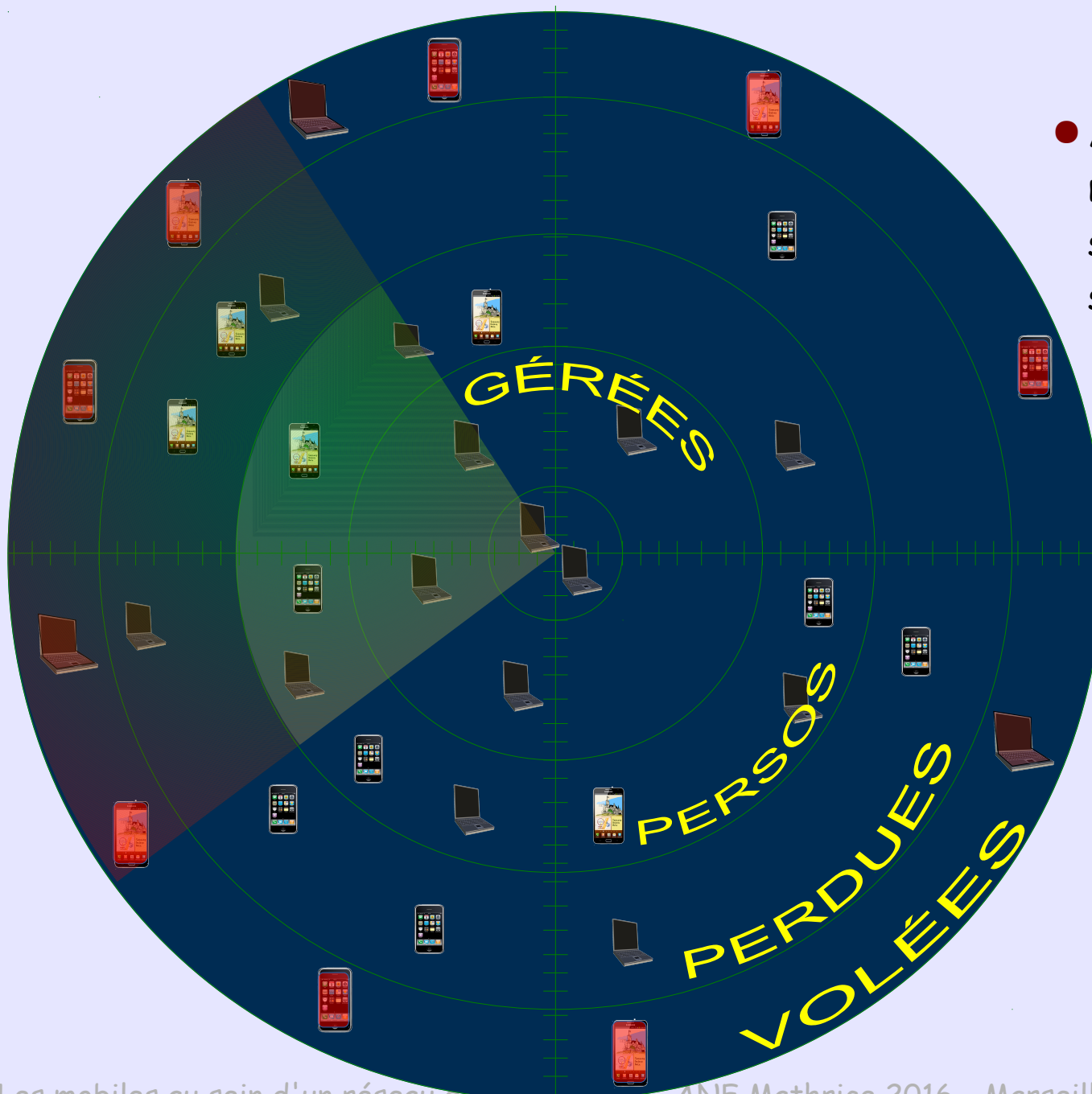


# Qu'est-ce qu'une machine perso (dite BYOD)

- C'est une machine non-gérée, donc **inconnue**
- Impossible de connaître son vécu (perdue, volée, recyclée ...)  
Incidents **pas déclarés** au CSSI.
- Impossible de savoir **qui y a accès**
- Impossible de connaître le **niveau de sécurité** (verrouillage d'écran par exemple, anti-virus...)
- Le propriétaire devient **de plus en plus indépendant** et s'éloigne d'un support proche.



# Machine perso : cette inconnue



- A terme, la plupart des machines qui auront accès au système d'information seront totalement inconnues.

## Rappel PSSI-E

*EXP-MAIT-MAT : maîtrise des matériels. Les postes de travail - y compris dans le cas d'une location - sont fournis à l'utilisateur par l'entité, gérés et configurés sous la responsabilité de l'entité. **La connexion d'équipements non maîtrisés**, non administrés ou non mis à jour par l'entité (qu'il s'agisse d'ordiphones, d'équipements informatiques nomades et fixes ou de supports de stockage amovibles) **sur des équipements et des réseaux professionnels est interdite.***

# Machine perso + Services en ligne

---

- La mise en ligne de services avec des authentifications **basées sur de simples mots** de passe favorise l'**usage incontrôlé** de machines perso.
- Favorise la **dispersion** des mots de passe dans une **multitude** d'appareils.
- Avoir accès à de tels matériels (piratage, sphère privée) c'est la certitude d'y trouver des identifiants, y compris professionnels.
- Les données sont dupliquées dans un environnement qui n'a plus aucune sécurité

# L'influence sociétale

L'influence de l'entourage et plus largement de la société aura de plus en plus d'influence sur le niveau sécurité du système d'information.

## Exemples

- Bons conseils d'un copain qui n'a pas les mêmes critères de sécurité.
- Mauvaises pratiques par mimétisme (les autres le font bien, alors pourquoi pas moi?)
- Suivre la dernière mode (applications/jeux médiatisés qui se répandent partout)
- Publicités et influences des différents acteurs
- etc



# Exemples simples de mauvaises pratiques

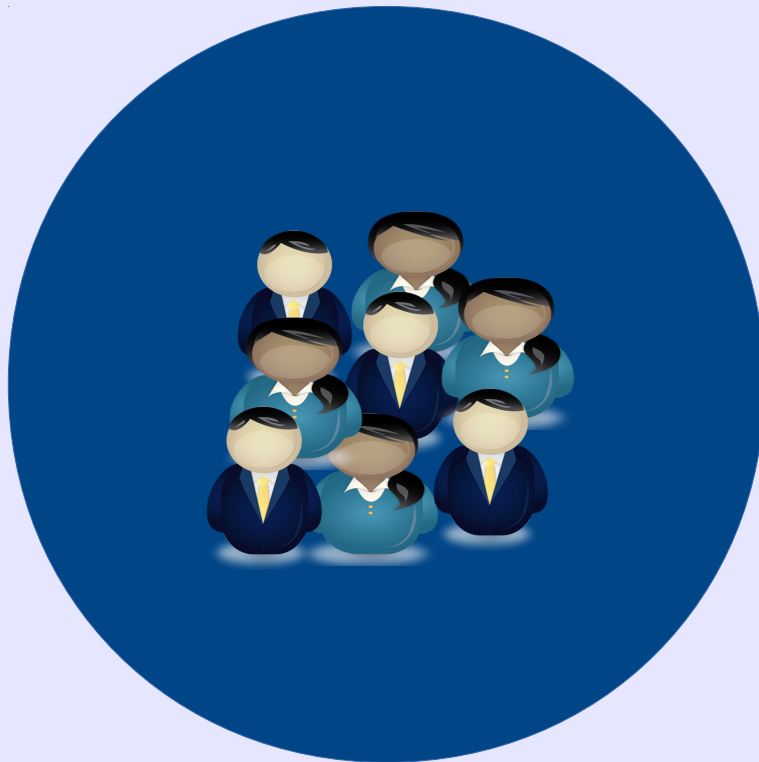
---

- Ne pas avoir de verrouillage d'écran
- Mettre le smartphone dans la poche arrière de son pantalon
- Poser le smartphone sur la table
- Partager consciemment ses mobiles avec de tierces personnes
- Partager inconsciemment ses mobiles
- Installer toutes sortes d'applications « pour voir »
- Installer des applications dont le mode de fonctionnement est dangereux.

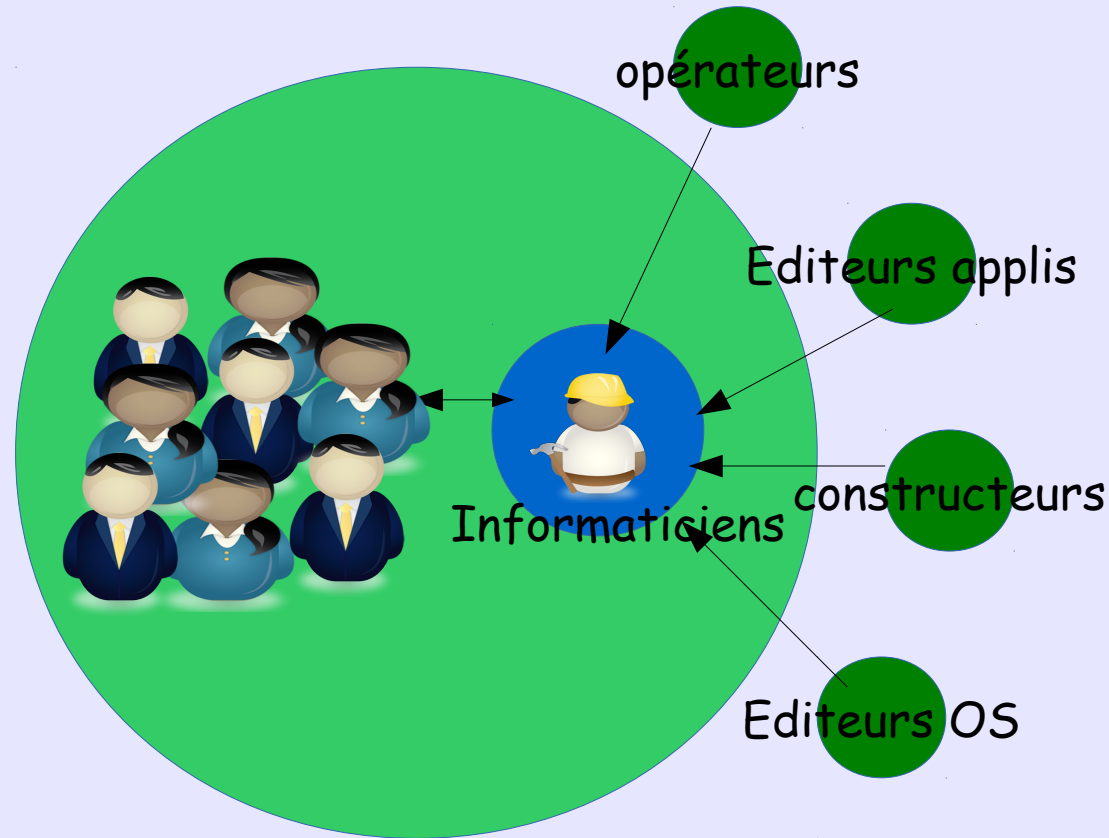


# L'éloignement du support professionnel

Avant,  
le monde était simple....tout passait par les informaticiens



Sphère privée

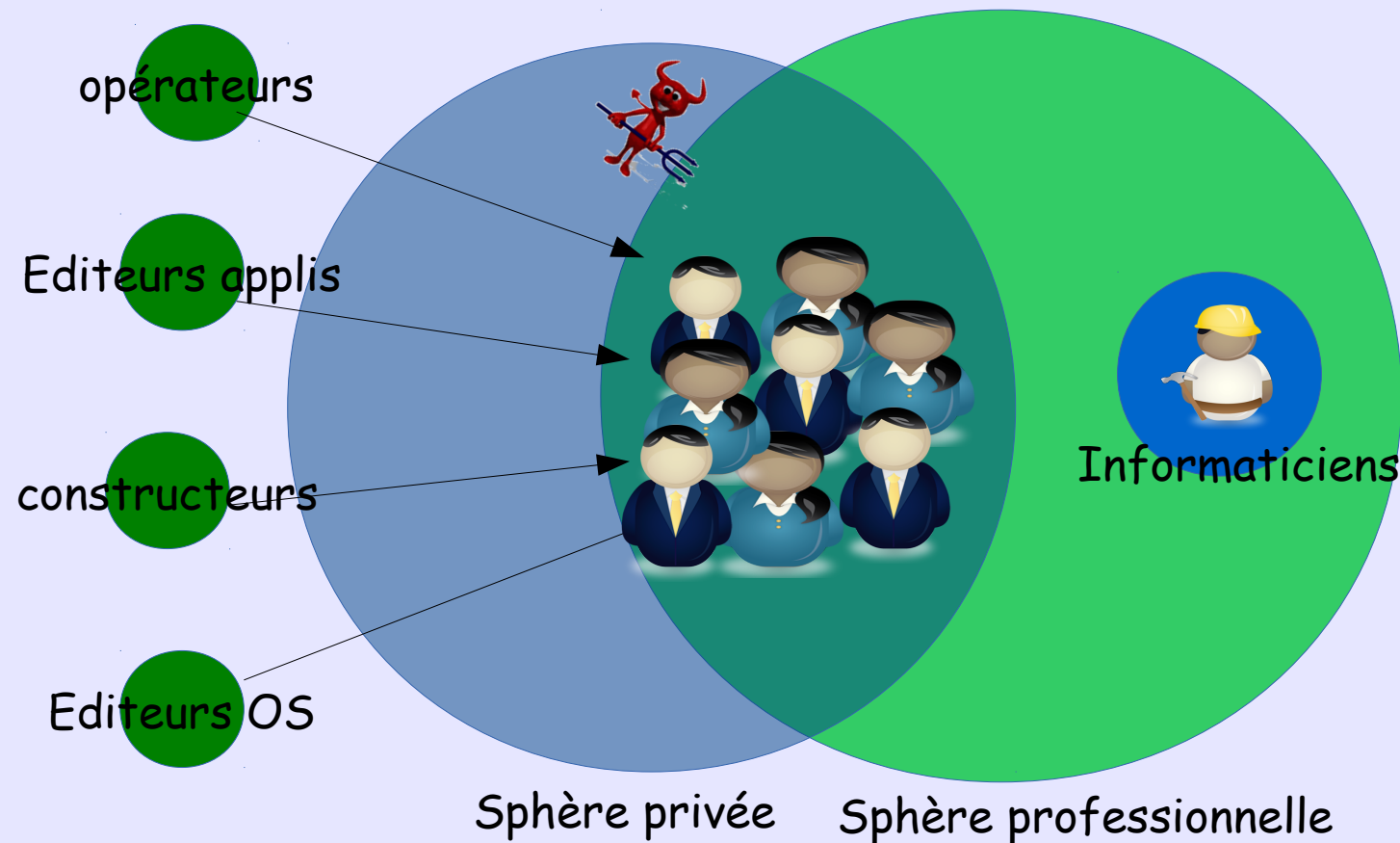


Sphère professionnelle

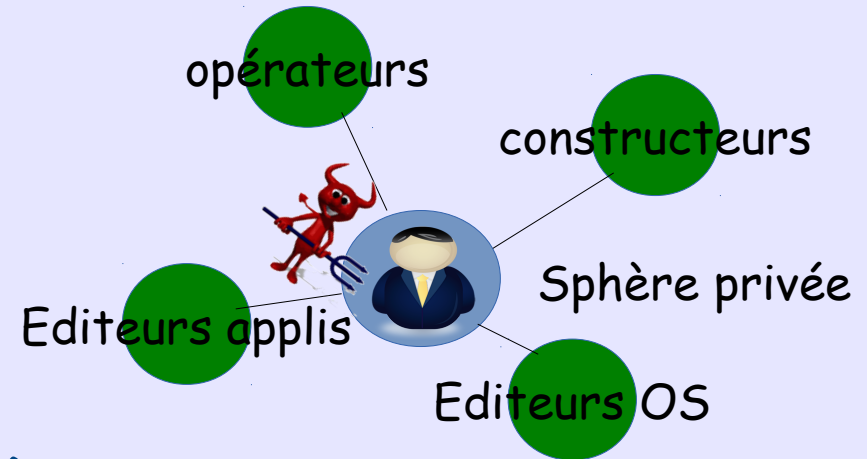
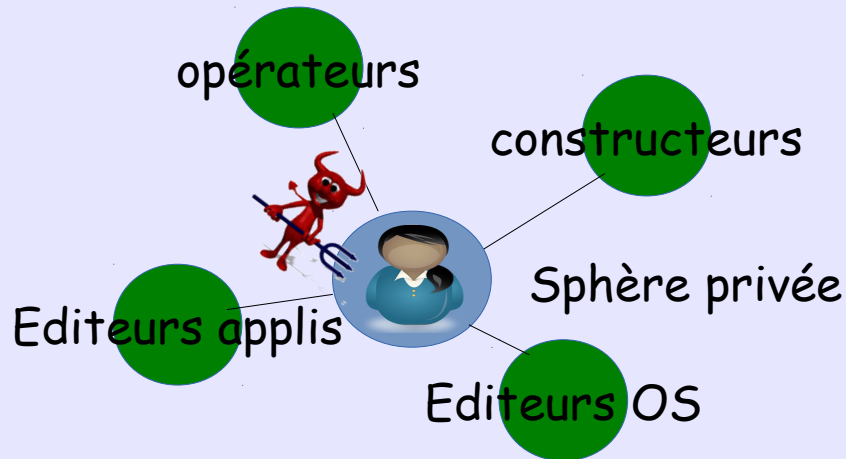


# L'éloignement du support professionnel

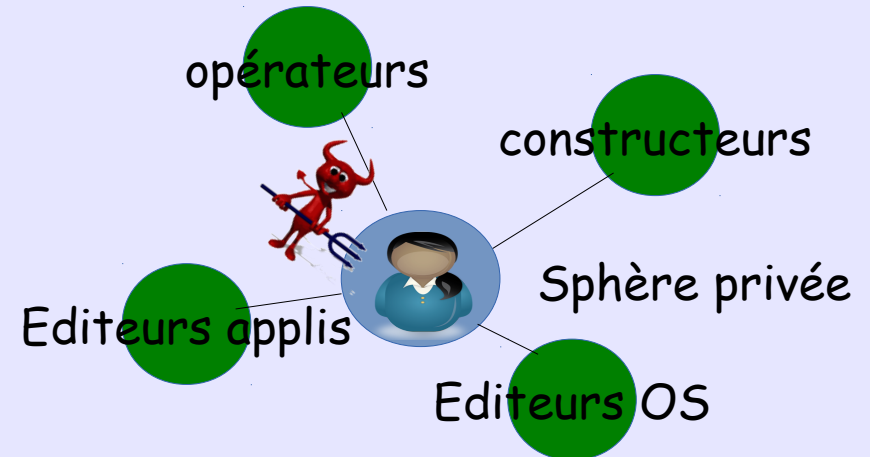
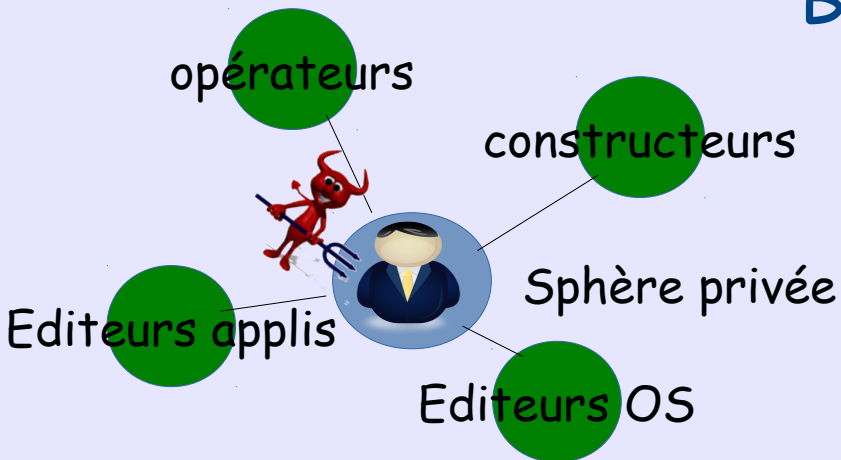
Aujourd'hui,  
tous les acteurs s'adressent directement aux utilisateurs



# L'éloignement du support professionnel



BIENTÔT ?





# Le modèle de sécurité mis à mal par le BYOD ( Sauvage )

Modèles de sécurité	Réalité
Confidentialité du mot de passe Ex : Ne coller pas votre mot de passe sur votre clavier	Mot de passe stocké dans de multiples endroits, de grandes chances d'être compromis au moins dans la sphère privée.
Confidentialité des informations Ex : On sait où sont les données avec des droits d'accès.	Informations synchronisées dans de nombreux endroits sans aucune garantie de confidentialité.
Chiffrement des ordinateurs	Données synchronisées dans de nombreux endroits, pas chiffrés ou pas protégés.
Incidents de sécurité connus	Incidents de sécurité ne sont pas remontés, voire même pas connus de l'utilisateur. Ex : Par de gestion de parc, pas de logs pour détecter les problèmes...
Déclarations d'incidents obligatoires	Déclarations d'incidents impossibles.
Analyse des compromissions possibles	Analyse des compromissions impossibles.

## Le BYOD c'est comme la fin des uniformes dans les écoles

Citation magazine MISC N°66 mars/avril 2013



# II

---

## Architecture des systèmes d'exploitation des mobiles

---



# Processeurs

---

- Les processeurs des mobiles (et autres systèmes embarqués) doivent être économes et ne pas chauffer.
- D'où l'émergence des processeurs de type ARM qui équipent tous les mobiles.
- ARM Ltd (Advanced RISC Machines) est la société qui a conçu le processeur du même nom, mais qui ne les fabrique pas. Elle vend uniquement des licences aux fondeurs.
- La plupart des mobiles récents ont des processeurs déclinés de la famille ARM Cortex.
- Samsung => Exynos  
Apple => Ax  
Qualcomm => Snapdragon
- Intel pousse son architecture atom, compatible X86

---

# Architecture générale d'iOS

---



# Modèle en couches



- iOS Dérive de MACOS X
- Le système est organisé en 4 couches
  - **Core OS** : Contient les fonctions de bas niveau
  - **Core service** : Fournit les services de base aux applis
  - **Media** : Fournit les fonctions multimedia aux applis
  - **Cocoa touch** : Interface pour créer des applis



# Démarrage du système

---



Il y a 3 modes de démarrage du système

- Normal
- Recovery
- DFU (Device Firmware update)

# Démarrage du système : Mode normal

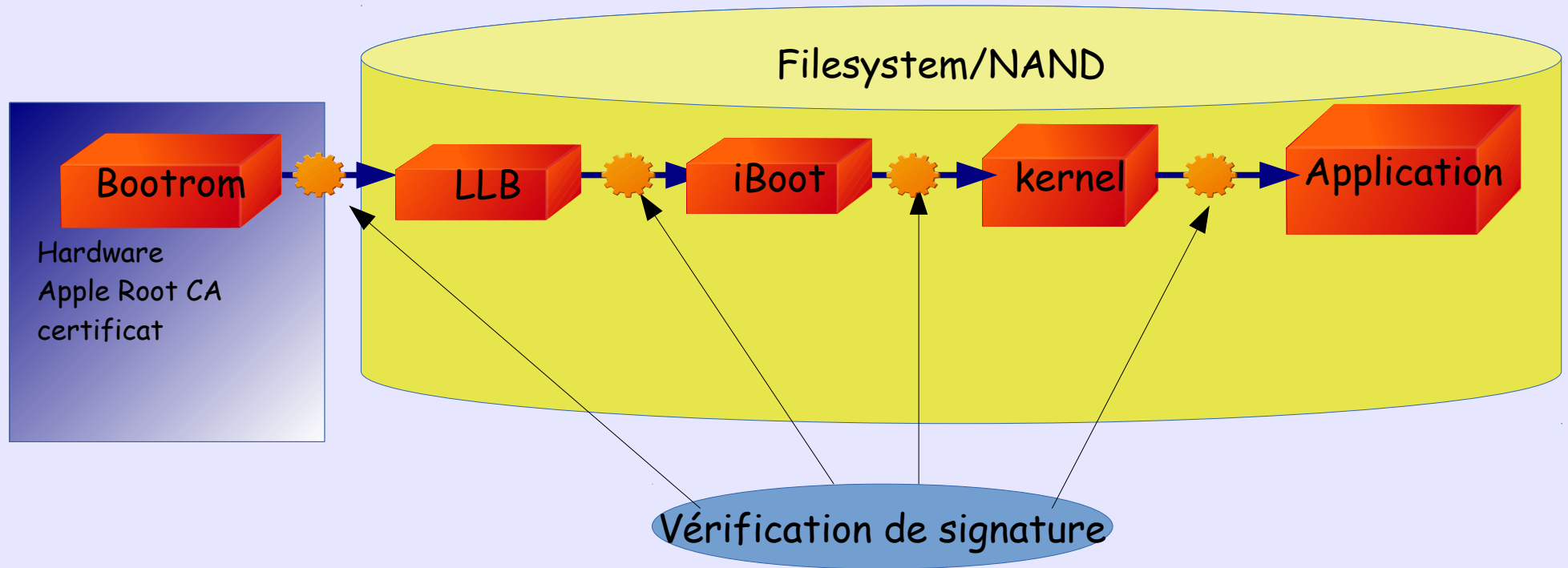


- Le démarrage initial est réalisé depuis la ROM
- Entre chaque étape l'intégrité du code est vérifiée (signature cryptographique)
- La bootrom contient la clé publique de l'autorité de certification Apple qui permet de vérifier l'intégrité du code LLB qui réside en mémoire flash (de type NOR, lecture rapide, écriture lente)
- L'étape LLB (Low Level Bootloader) fait de même avec le code iBoot avant de l'exécuter
- iBoot fait de même avec le kernel avant de le lancer (le kernel est sur de la mémoire NAND, écriture rapide, lecture lente)
- Si la vérification de signature échoue :
  - Entre Bootrom et LLB => Le mobile démarre en mode DFU
  - Après => Le mobile démarre en mode recovery





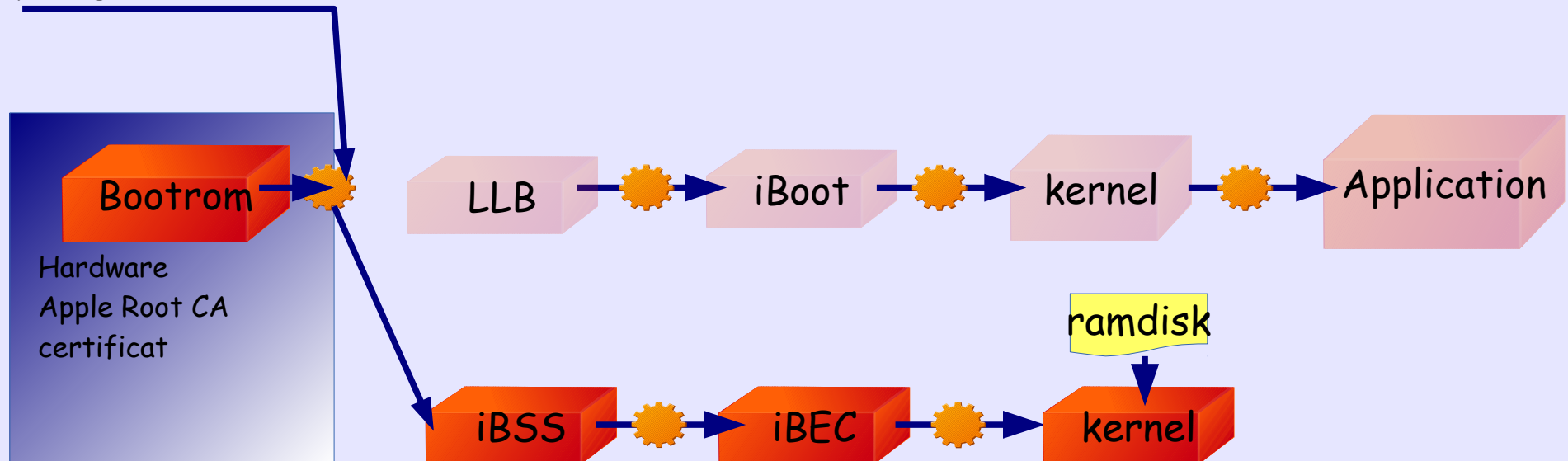
# Démarrage du système : Mode normal



# Démarrage du système : Mode DFU (Device Firmware Update)

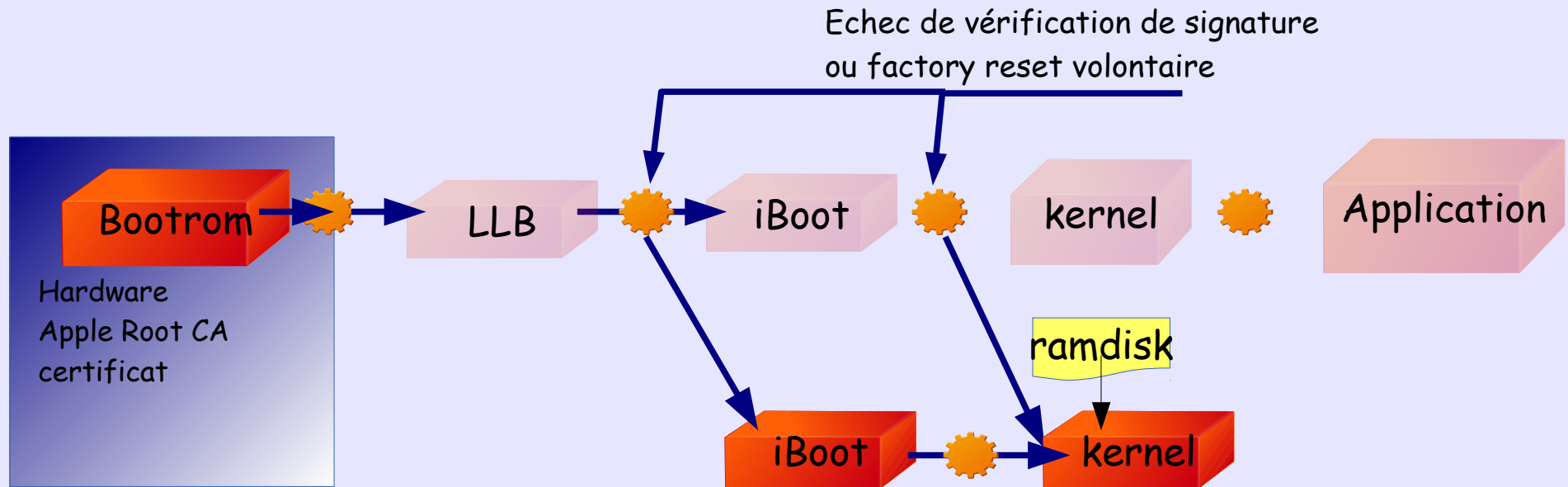


Echec de vérification de signature  
ou passage volontaire



- Utilisé pour mettre à jour le firmware depuis iTunes sur un ordinateur
- Permet de choisir le firmware à installer (revenir en arrière éventuellement)
  - ✓ On éteint le mobile et on le connecte via USB sur l'ordinateur sur lequel on lance iTunes.
  - ✓ On appuie simultanément sur les boutons sleep et Home pendant 10sec. Puis on relâche le bouton sleep et on maintient Home jusqu'au message de connexion de iTunes.

# Démarrage du système : Mode RECOVERY



Le mode recovery force une mise à jour sur la dernière version du firmware



## C'est quoi ?

- Le Jailbreak a pour but de contourner les sécurités Apple afin d'obtenir les privilèges root et prendre le contrôle de l'appareil.
- Conséquence : supprime la plupart des sécurités et rend l'appareil très vulnérable.
- Cela n'a rien à voir avec le déverrouillage de la carte SIM

## A quoi ça sert ?

- Prendre le contrôle de l'appareil pour en extraire ses secrets
- Installer des applications à partir d'autres dépôts que celui d'Apple => risques virus
- Installer des applications payantes gratuitement => illégal

## Comment ?

En exploitant une (ou plusieurs vulnérabilités), notamment dans la chaîne de boot



- Lorsqu'un appareil est jailbreaké un **serveur SSH est installé**
- Mot de passe par défaut « alpine » 
- Scan pour rechercher des iOS jailbreakés

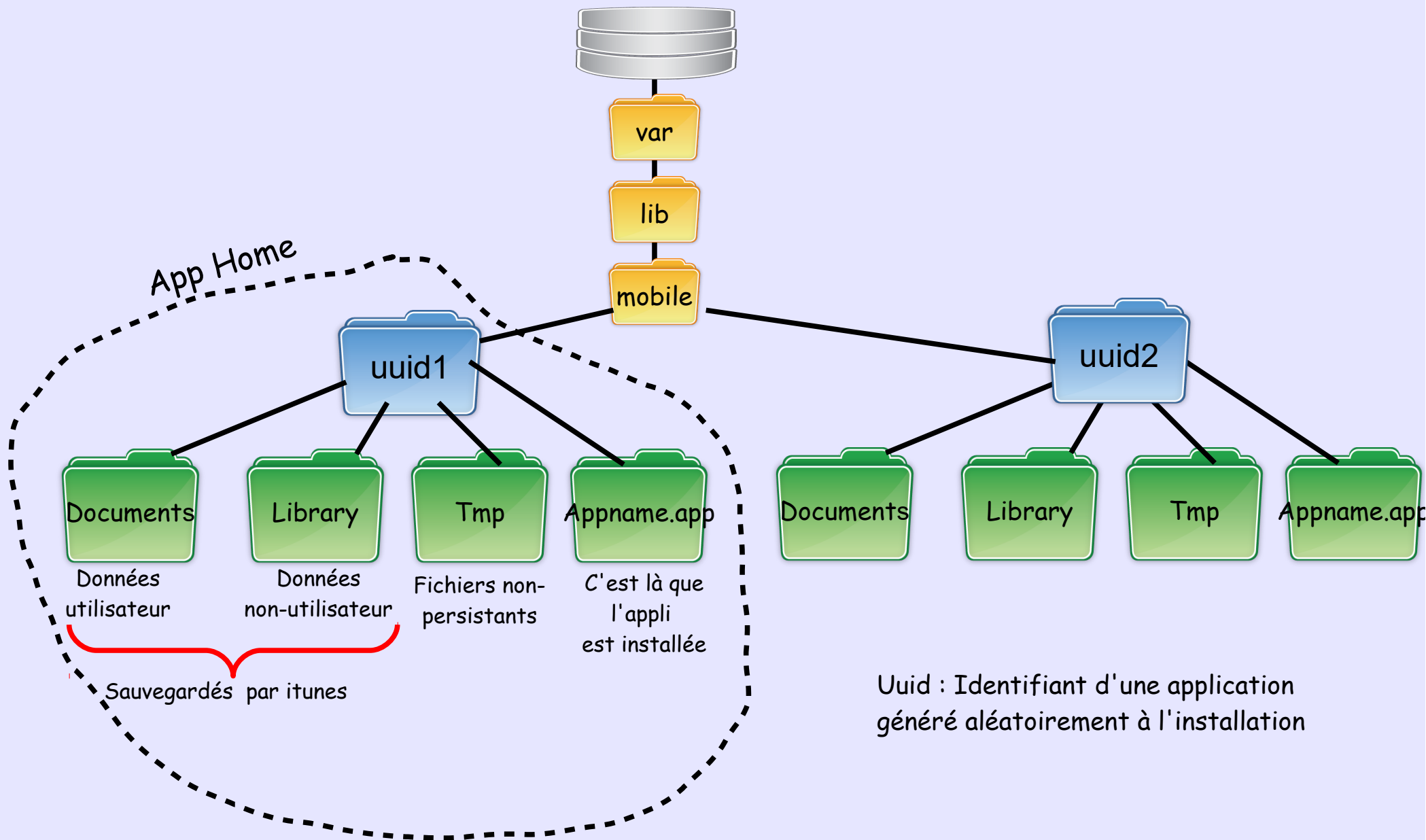
**sshpass -p alpine ssh -o StrictHostKeyCheck**

- Utilisé par des malwares (Ikee.B)



- Le système de fichiers est basé sur le système de fichier Unix (HFS+)
- Organisé en 2 partitions : root (ro) et data (rw)
- Lors de l'installation, chaque application se voit attribuer un répertoire **HOME** qui contient tout ce que l'appli a besoin. Elle ne peut accéder directement qu'à cet espace.
- Toutes les applications tournent sous le compte « **mobile** »
- Une application n'a pas accès au Home d'une autre application

# Système de fichiers iOS





- Il n'y a pas de stockage partagé entre les applications
- Un même fichier doit être dupliqué pour être accessible par deux applications
- Pour copier des fichiers d'un ordinateur vers iOS il faut
  - Qu'iTunes soit installé sur cet ordinateur
  - Avoir un compte Apple
  - Autoriser le mobile dans le compte Apple
  - On ne peut copier des fichiers de l'ordinateur => mobile que d'applications déjà existante sur le mobile.
- L'autre alternative consiste à faire des téléchargements par le réseau (serveur Web, cloud...)
- L'utilisateur n'a pas accès à l'arborescence



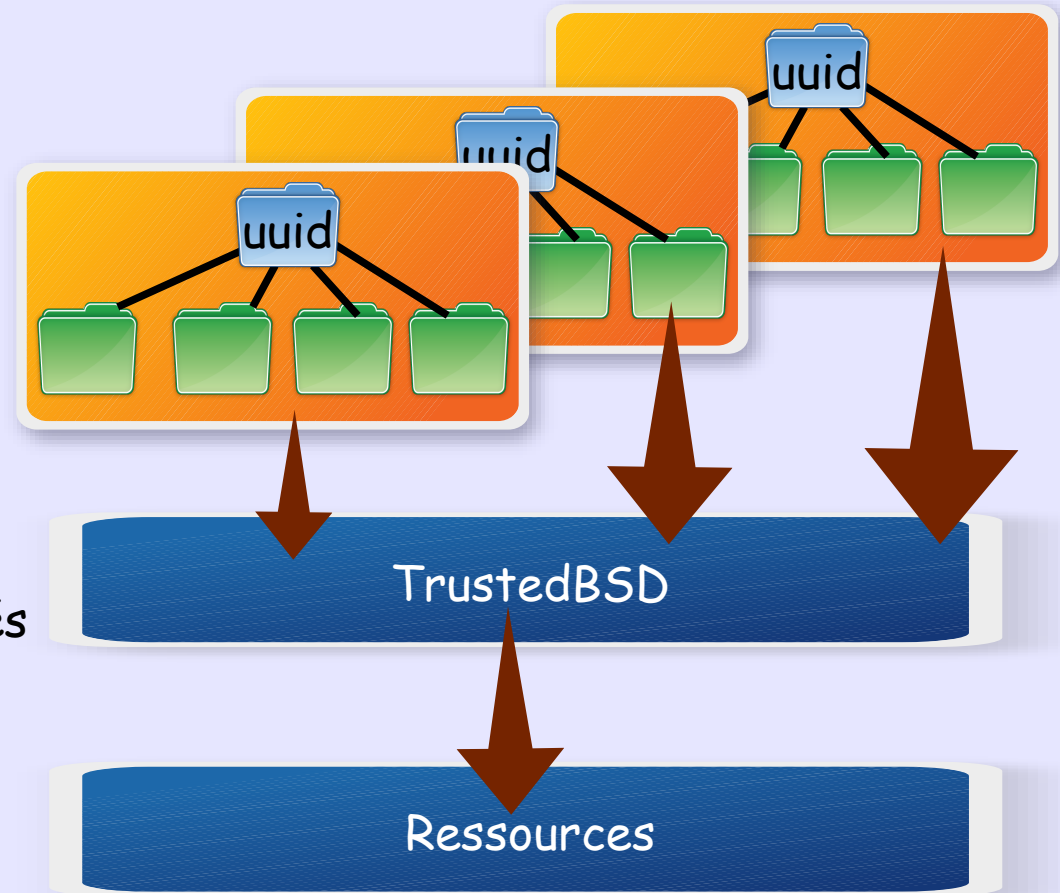
---

# Applications et permissions sous iOS

---

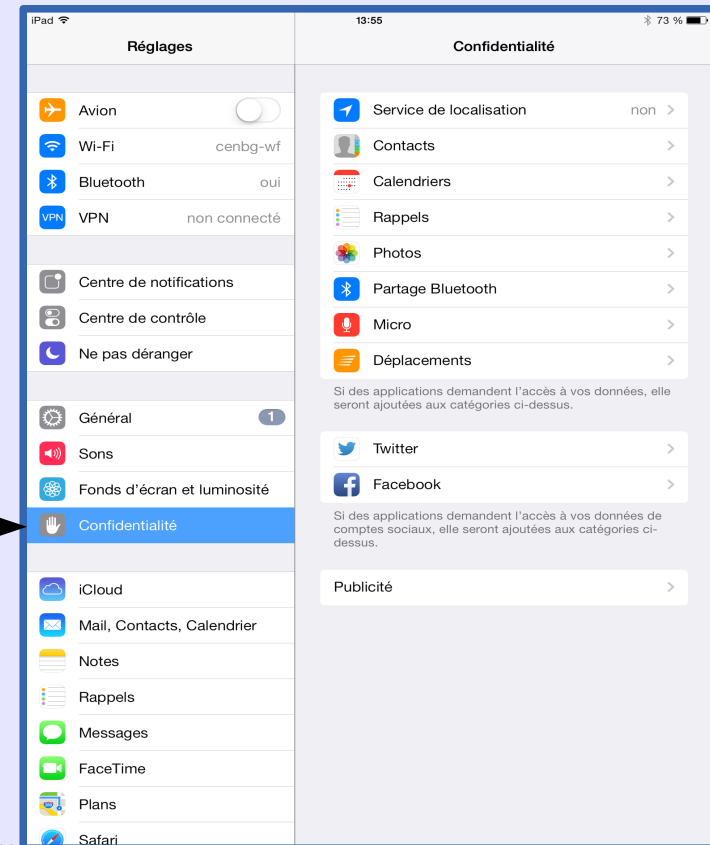
# Permissions sous iOS - Sandboxing

- Toutes les applications tournant sous le même userid (**mobile**), les mécanismes de sécurité de base d'Unix ne sont pas suffisant pour assurer le cloisonnement des applications
- Le **sandboxing** est un mécanisme permettant de restreindre les possibilités d'un processus
- Basé sur **TrustedBSD** (équivalent à SELINUX)
- Lors de l'installation d'une appli le système crée une sandbox et lui affecte les privilèges



# Permissions sous iOS - Sandboxing

- Lors de l'installation d'une application l'utilisateur ne voit pas quelles ressources demande l'application.
- C'est le processus de validation d'Apple qui décide si ces ressources sont légitimes (600 à 800 applications par jour...)
- Avant iOS 6 les applications tierces pouvaient avoir accès à des informations privées du type carnet d'adresses, aux informations de localisation, au N° de téléphone, aux configurations mail et Wifi, aux photos, au cache du clavier.
- Depuis iOS6, Lorsqu'une application veut accéder à ce type de données elle doit obligatoirement demander l'autorisation à l'utilisateur.
- On peut voir quelles applis accèdent à quels types de données. (réglages/confidentialité)



---

## Chiffrement des mobiles iOS

---





- Tous les matériels iOS possèdent un co-processeur cryptographique (secure enclave)
- Ce processeur contient deux clés (AES 256 bits)
  - UID key : spécifique et unique au matériel gravée dans le processeur à la fabrication
  - GID key : identique pour tous les processeurs du même type
- Ces clés ne peuvent pas être extraites du processeur.
- Les fichiers chiffrés ne peuvent être déchiffrés que sur le même matériel (là où se trouve l'UID utilisé pour les chiffrer)



- Introduit à partir de iOS 4.0
- Chaque fichier est chiffré avec une clé de chiffrement spécifique (per-file key)
- Des API interfacent les applications avec le système pour lui indiquer dans quelles circonstances les fichiers peuvent être déchiffrés
- C'est donc le développeur d'une application qui définit le niveau de protection de ses fichiers et pas le système
- L'application doit indiquer ce niveau au moyen d'une classe de protection.



## Exemple de classes de protection

### **NSFileProtectionNone**

Le fichier n'est pas protégé et peut toujours être accédé.

### **NSFileProtectionComplete**

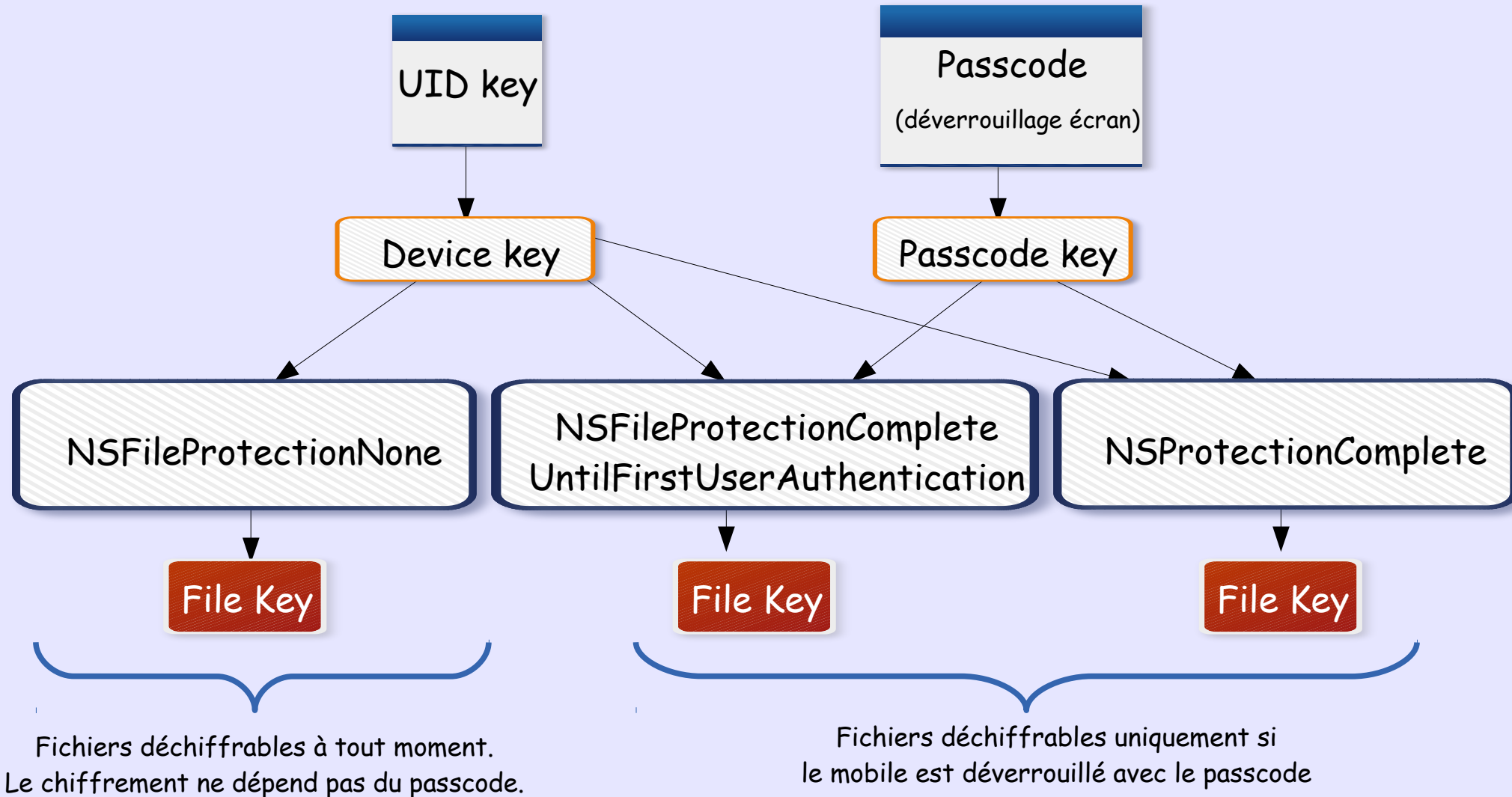
Le fichier est protégé et il est accessible uniquement quand le mobile est déverrouillé par le code utilisateur

### **NSFileProtectionCompleteUnlessOpen**

Le fichier ne peut être ouvert que lorsque le mobile est déverrouillé par le code utilisateur mais il peut rester ouvert lorsque le mobile est ensuite verrouillé (applications en arrière-plan)



## Hierarchie des clés de chiffrement







Oui mais ...

- C'est le développeur de chaque application qui décide du niveau de chiffrement
- L'utilisateur peut avoir le sentiment que tous ses fichiers sont protégés alors que ce n'est pas le cas

# Chiffrement des mobiles iOS : Keychain

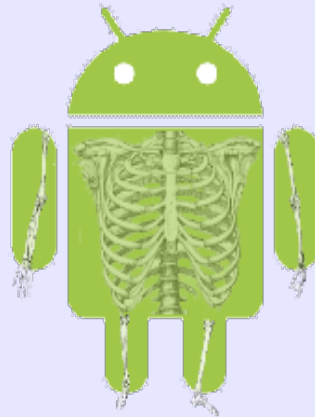


- Le **keychain** est une base de données SQLite qui permet de stocker des secrets du système d'exploitation ou d'applications tierces.
- On peut y stocker des mots de passe et des certificats (clé privée). ces informations sont chiffrées en appliquant une structure de classes de protection.
- Une application n'a accès qu'à ses propres Items.
- Un backup sauvegarde les secrets en conservant le chiffrement. Le mot de passe de déverrouillage n'est pas sauvegardé.
- L'utilisation du Keychain est à la charge du développeur (Keychain Services Task)

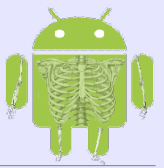
---

# Architecture générale d'Android

---



# L'évolution d'Android



2009



Cup Cake  
1.5

2009



Donut  
1.6

2009



Eclair  
2.0

2010



Froyo  
2.2

2010



Gingerbread  
2.3

2011



Honeycomb  
3.0

2011



Ice Cream  
Sandwich  
4.0

2012



Jelly Bean  
4.1-4.3.1

2013



KitKat  
4.4

2014



Lollipop  
5.0

2015

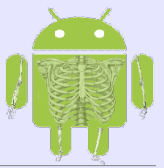


Marshmallow  
6.0

2016

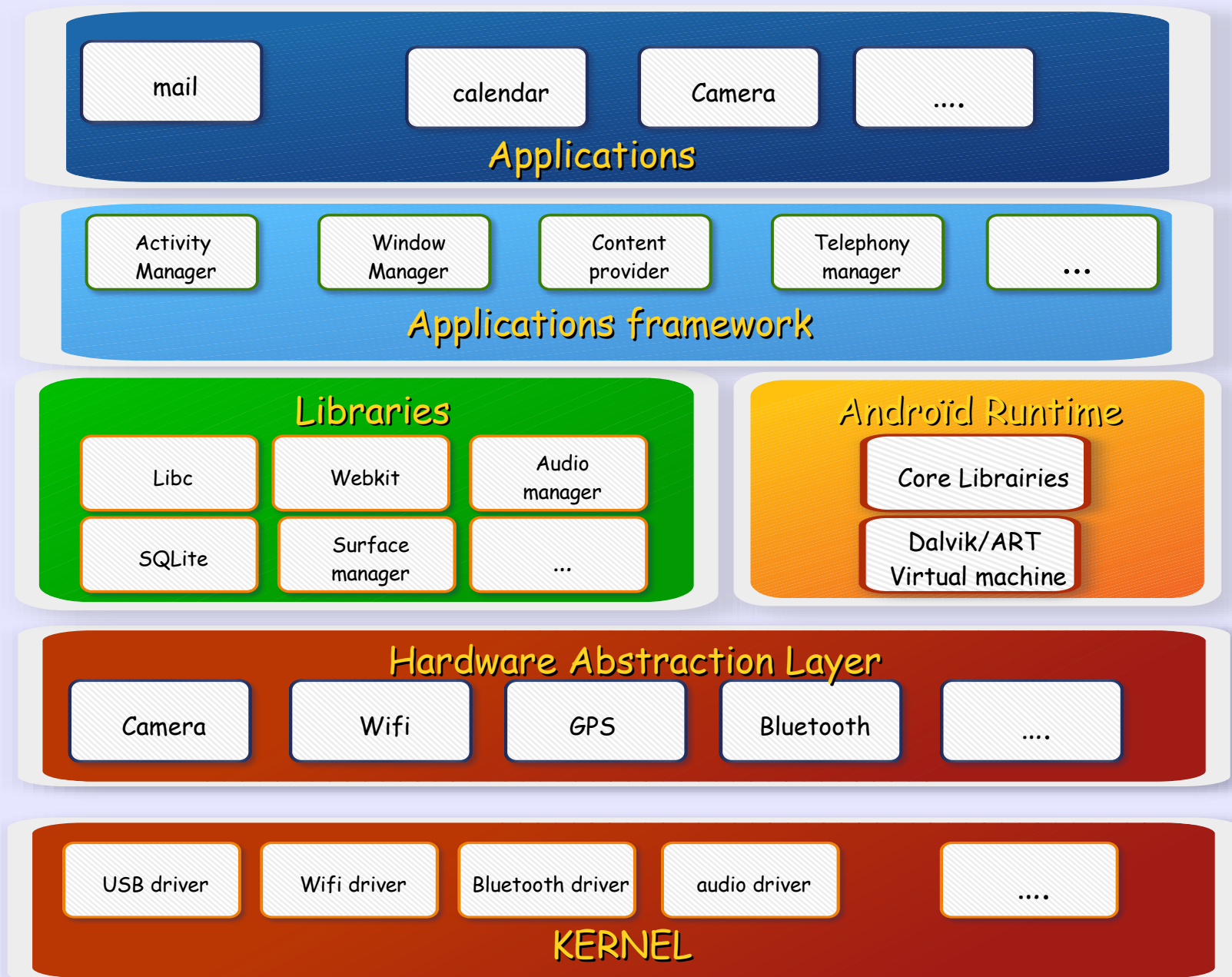
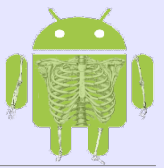


Nougat  
7.0



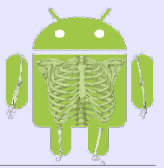
- Android est basé sur un noyau Linux
- Revu par Google pour tenir compte des spécificités des mobiles
  - performance CPU
  - Moins de mémoire
  - Besoin d'économiser la batterie
- Ce n'est pourtant pas un système GNU/Linux  
Google a beaucoup modifié le contexte GNU pour ses propres besoins
  - La librairie C s'appelle **Bionic**
  - La gestion des fenêtres n'est pas X-Window mais **Surface Flinger**  
(les constructeurs rajoutent éventuellement une surcouche telle que TouchWiz de Samsung)
- Le code source est disponible dans le projet **Android Open Source Project**  
(AOSP - <https://source.android.com/>)

# Architecture générale



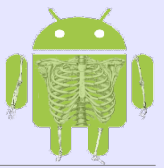


- Depuis la version 7.0 Android impose le "**verified boot**" (présent depuis 6.0)
- Vérification de l'intégrité de l'image de boot qui aurait pu être modifiée par un rootkit ou malware (vérification cryptographique)
- A partir d'une racine matérielle, chaque étape vérifie l'intégrité de l'étape suivante.
- La machine ne boote pas ou bien avec des capacités limitées, avec le consentement de l'utilisateur (pour récupérer des données par exemple)
- Avec 7.0 Google a rajouté un mécanisme de récupération des erreurs non-malicieuses (Forward Error Correction)



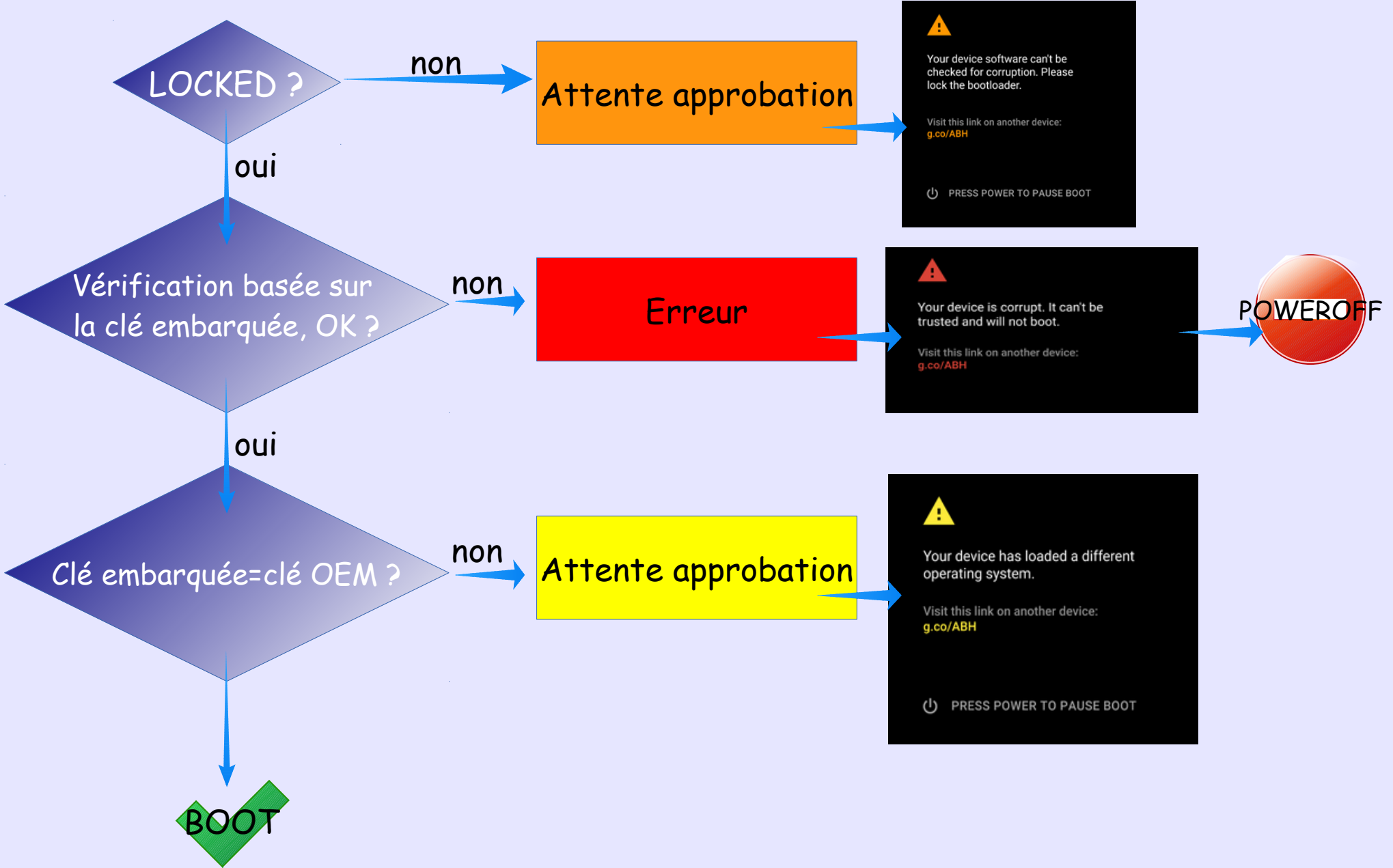
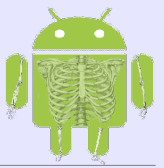
- **BootState** : Définit le niveau de protection du boot
  - **GREEN** : Toute la chaîne de boot a été vérifiée
  - **YELLOW** : La partition Boot a été vérifiée, mais attend une validation
  - **ORANGE** : L'appareil peut-être flashé après validation
  - **RED** : La vérification a échoué, l'appareil s'arrête.



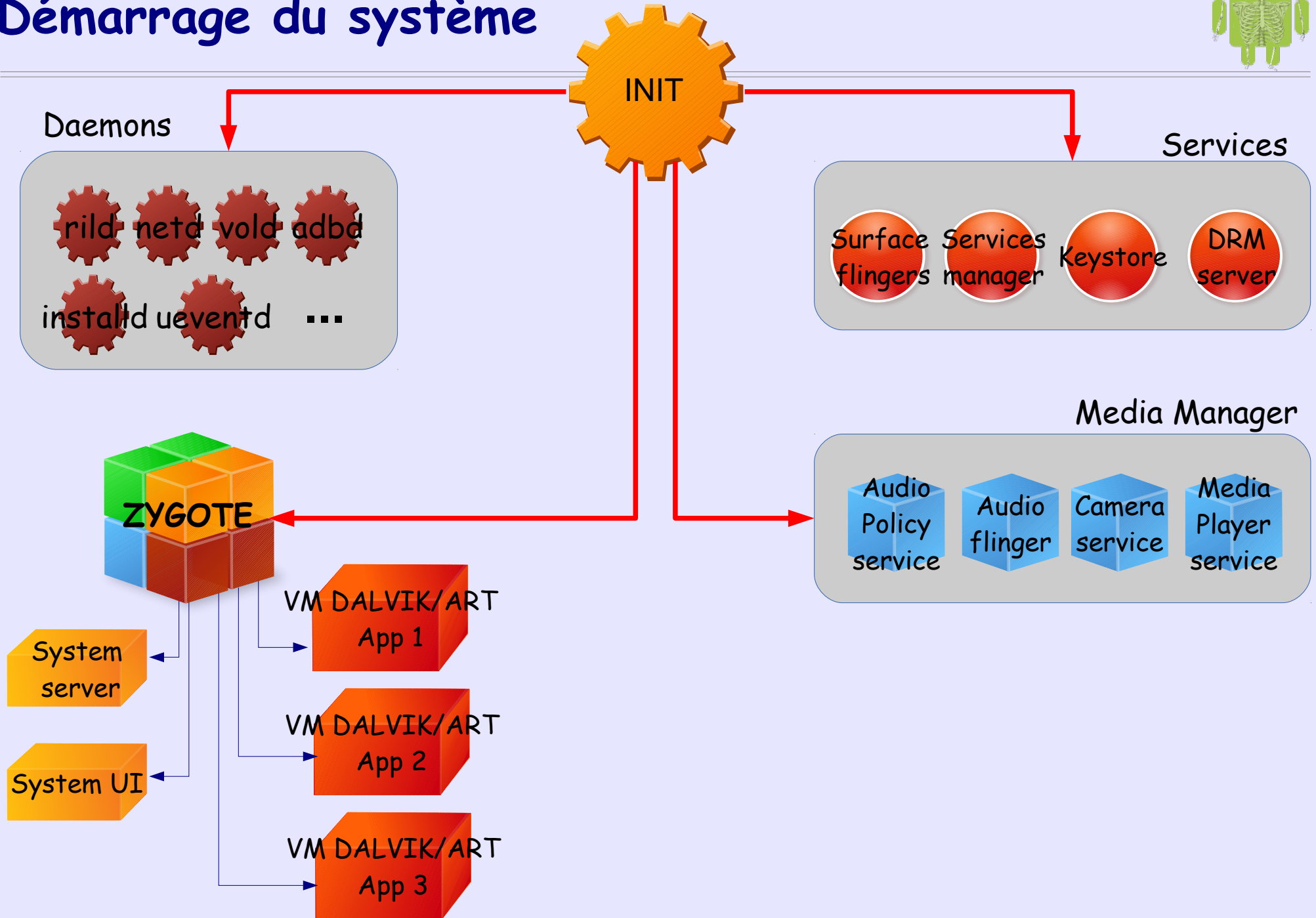
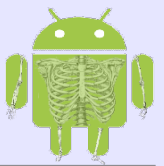


- **DeviceState** : Deux états possibles **LOCKED** ou **UNLOCKED**
  - **LOCKED** ➞ L'appareil ne peut pas être flashé (Bootstate= GREEN ou RED)
  - **UNLOCK** ➞ L'appareil peut être flashé (Bootstate =ORANGE)
- **Classes** : Les fabricants peuvent définir deux types d'appareils :
  - CLASS A** ➞ Ceux qui ne permettent pas de flasher
  - CLASS B** ➞ Ceux qui permettent de flasher

# Démarrage du système



# Démarrage du système



---

# Environnement des applications Android

---

# Machine virtuelle DALVIK/ART

---

- Les applications sont écrites en Java.
- Les applications tournent dans des machines virtuelles Java, indépendamment du hardware.
  - Anciennement **DALVIK** et depuis Android 5.0, **ART** (Android RunTime)
- Le code source est compilé en **byte-code Dalvik** (DEX) qui est exécuté par la machine DALVIK/ART (DEX=Dalvik Executable)
- Il est possible de coder en C/C++ en utilisant JNI (Java Native Interface)
- ART est complètement compatible avec les fichiers DEX

# Machine virtuelle DALVIK/ART

---

- Principale différence entre DALVIK et ART

- **DALVIK** utilise un procédé dit **Just-In-Time (JIT)**

Le code DEX est converti en instructions machine au moment où l'application en a besoin.

Avantages : usage mémoire réduit, taille de l'appli réduit sur le stockage.

Inconvénient : Exécution plus lente, plus de ressources processeurs consommées, réduction de l'autonomie.

- **ART** utilise un procédé dit **Ahead-of-Time (AOT)**

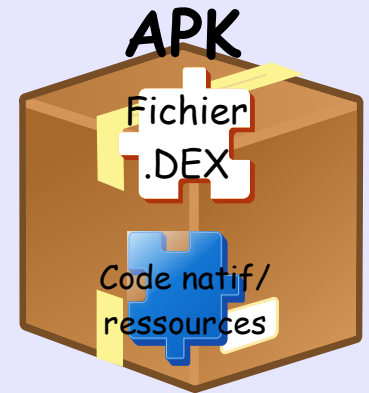
Le code DEX est converti en instruction machine au moment de l'installation.

Avantages : exécution plus rapide, moins de ressources processeurs, augmentation de l'autonomie

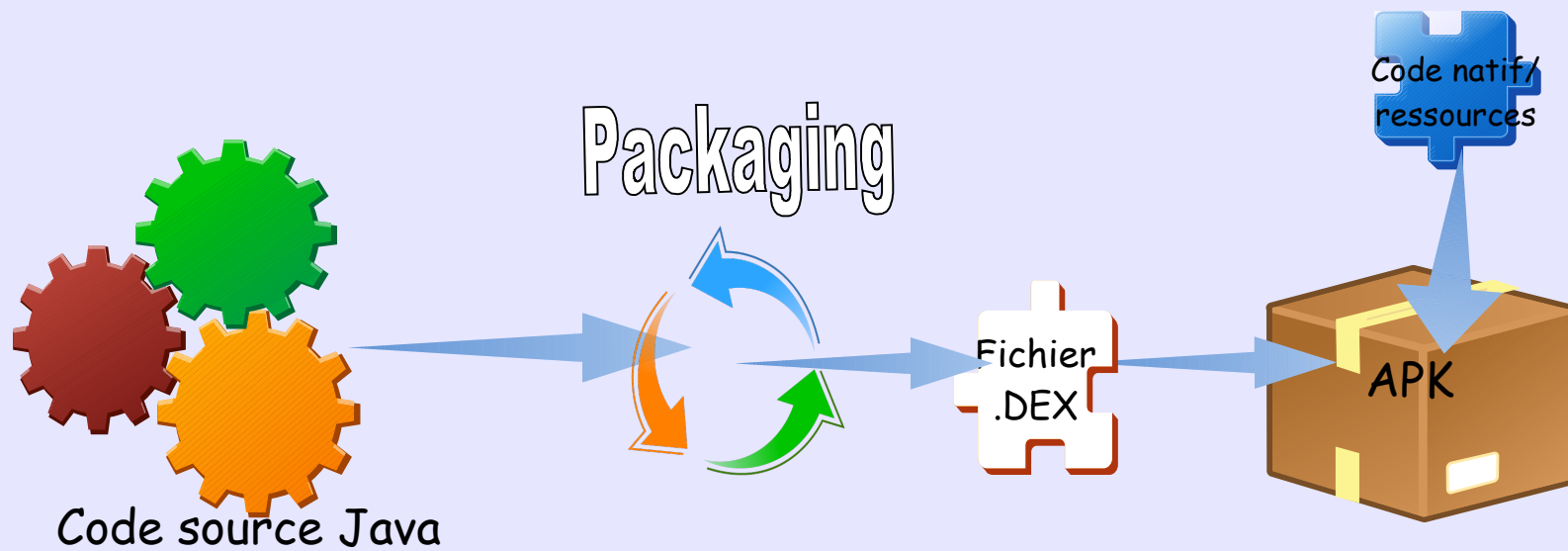
Inconvénient : Plus de mémoire, plus de stockage.

# Structure des applications

- Les applications sont packagées dans un format spécifique : **APK**
- Tous les fichiers nécessaires à l'appli sont dans le fichier APK
- Chaque application fonctionne sous un UID/GID qui lui est propre
- Les applications sont installées par le daemon installd qui est un processus qui tourne sous root (notamment c'est lui qui crée les UID et positionne le owner sur les fichiers)
- Les APK sont placés soit dans **/system/app** (applis pré-installées), soit dans **/data/app** (pour les applis téléchargées).

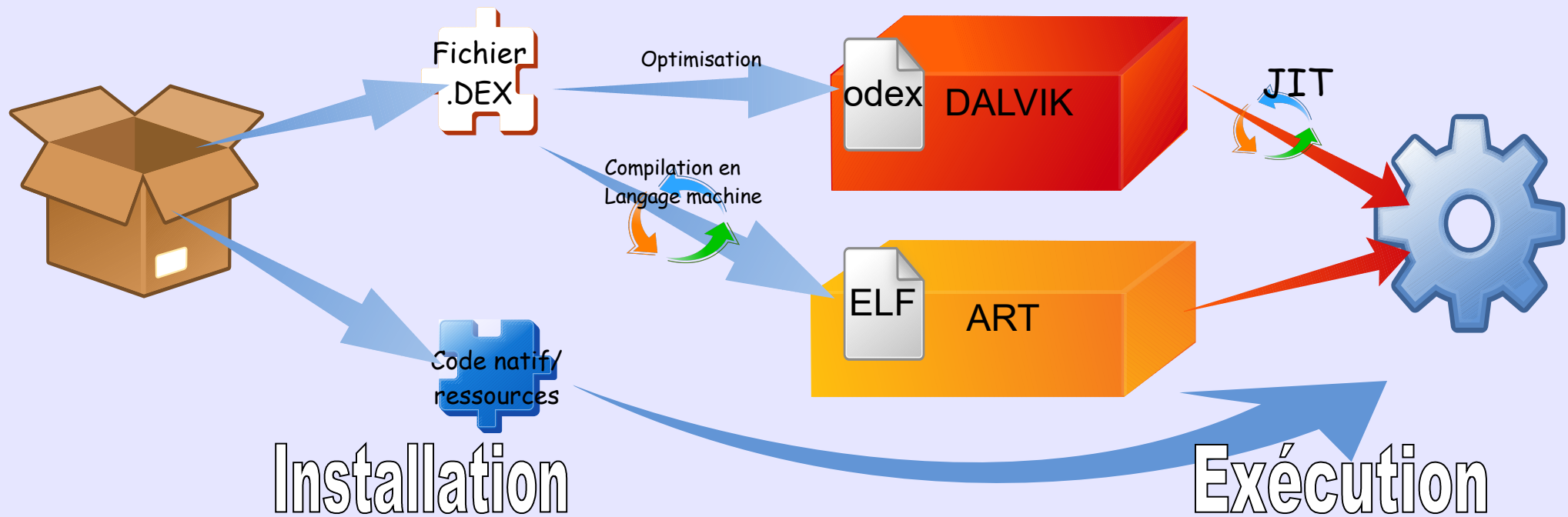


# Packaging





# Installation/exécution

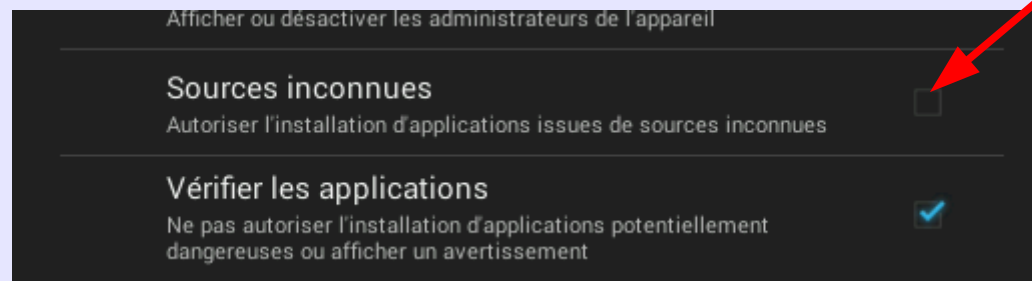


# Origines des applications

- Les applications sont normalement installées depuis **Google Play**
- Il est aussi possible d'installer des applis depuis d'autres sources :
  - Un fichier Apk
  - D'autres stores
- Root du système + origines douteuses des applis = la plus grandes causes de malwares sous Android.
- On peut y trouver des applis gratuites alors qu'elles sont payantes sur Google Play. Les installer constitue une violation de la licence et des droits d'auteur.
- Distribuer des APK d'applis payantes est aussi illégal

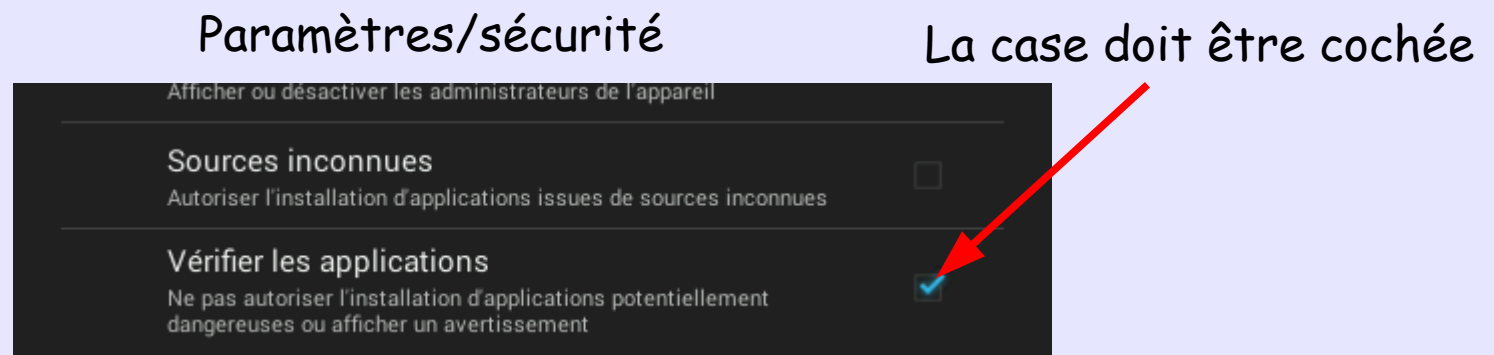
La case doit être décochée

## Paramètres/sécurité



# Protection native contre les malwares.

- Depuis 2012 Google effectue des vérifications sur les applications déposées dans Google Play au moyen d'un processus appelé **bouncer**
  - Recherche des malwares connus lorsque l'appli est copiée dans Google Play
  - L'appli est testée dans un environnement virtuel de Google pour rechercher les comportements malsains.
- Depuis 2013, l'option « **Vérifier les applications** » a été rajoutée dans les paramètres Android et permet de scanner une appli lors de son installation, quelle que soit la source d'installation.
- Depuis Mars 2014, cette option permet un scan permanent des applis installées



---

## Android sur PC : natif ou virtuel

---



# Android sur PC : natif ou virtuel

---

- Faire tourner un système Android sur PC en natif ou dans une **machine virtuelle**.
- Très proche d'un système réel
- Pour se **former**
- Faire des **démos**
- **Tester** des versions récentes d'Android ou des applications

## Les solutions existantes

### Virtualisation :

**Android x86** : Portage d'Android dans l'environnement x86  
Fonctionnement en machine virtuelle ou en natif

### En natif :

**Remix OS** : s'installe sur une clé USB. Interface graphique adaptée au PC

# Android sur PC : Android X86

---

- <http://www.android-x86.org/>
- Projet de portage d'Android sur les plate-formes X86
- Le projet fourni des images ISO de systèmes Android
- Ces **images** permettent :
  - une utilisation en Live sans installation
  - Installer le système sur un disque virtuel ou physique.
  - Ou sur une clé USB
- La connectivité réseau est disponible
- **Play Store** est pré-installé : Installation des applications comme sur un smartphone/tablette
- Passage en root très facile (ALT+F1/ ALT+F7)

# Android sur PC : Android X86

L'image ISO peut être utilisée soit pour booter en Live, soit pour installer le système sur disque

- Il faut d'abord créer la machine virtuelle (par exemple Virtual Box)

Exemple de configuration virtuelle :

1 Go de mémoire

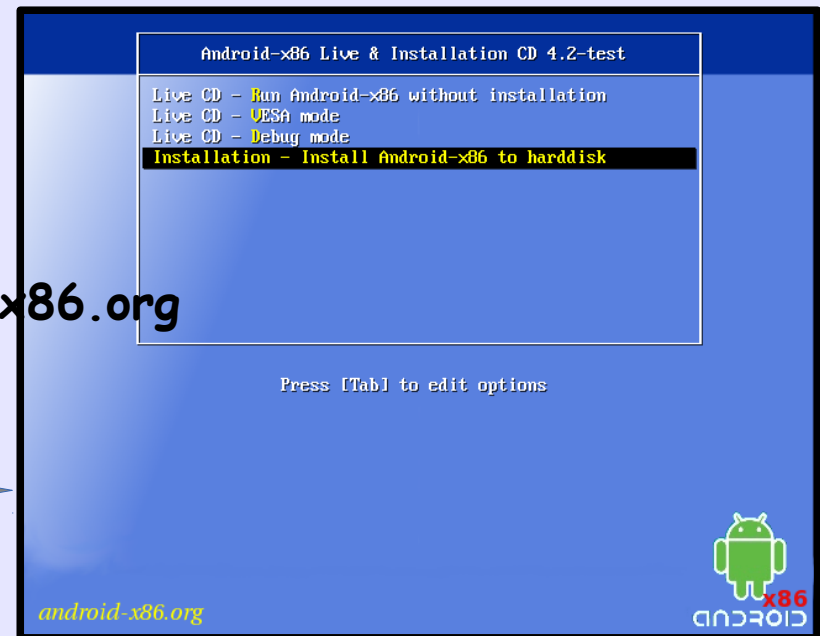
1 disque virtuelle 1Go (uniquement pour faire l'installation)

1 Lecteur CD/DVD qui pointe sur l'image ISO.

Option de démarrage : boot sur le CD/DVD

- Télécharger l'image ISO depuis le site [android-x86.org](http://android-x86.org)

- Booter la machine virtuelle



# Android sur PC : Android X86

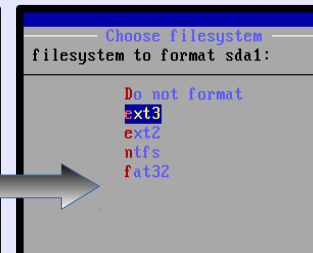
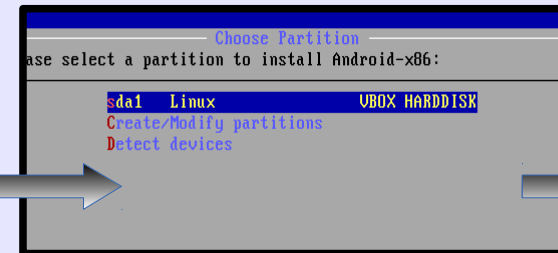
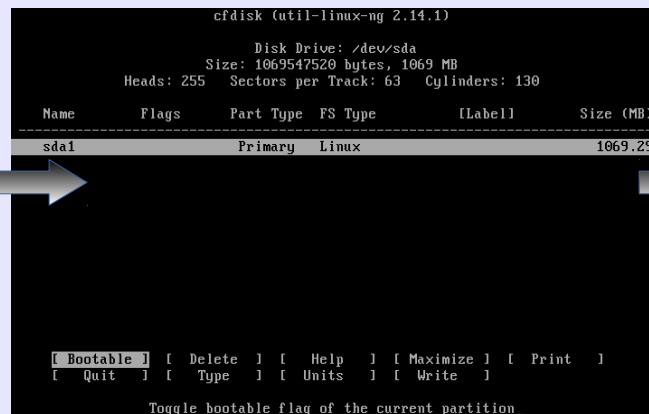
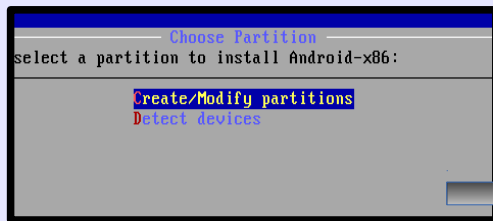
## Pour installer Android sur le disque virtuel

- Booter sur l'image ISO
- Choisir l'option « Install Android to harddisk »

*Choisir Create/Modify  
Pour créer la partition*

*Dans la fenêtre cfdisk créer  
une partition bootable sur la  
totalité du disque.*

*Choisir cette partition et la formater en ext3*

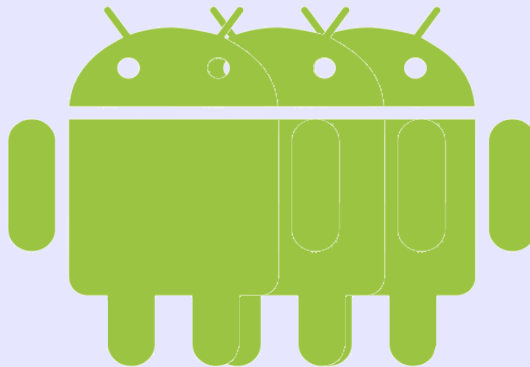




---

# Support multi-utilisateurs sous Android

---



# Support multi-utilisateurs sous Android<sup>4.2 / 4.3 +</sup>

---

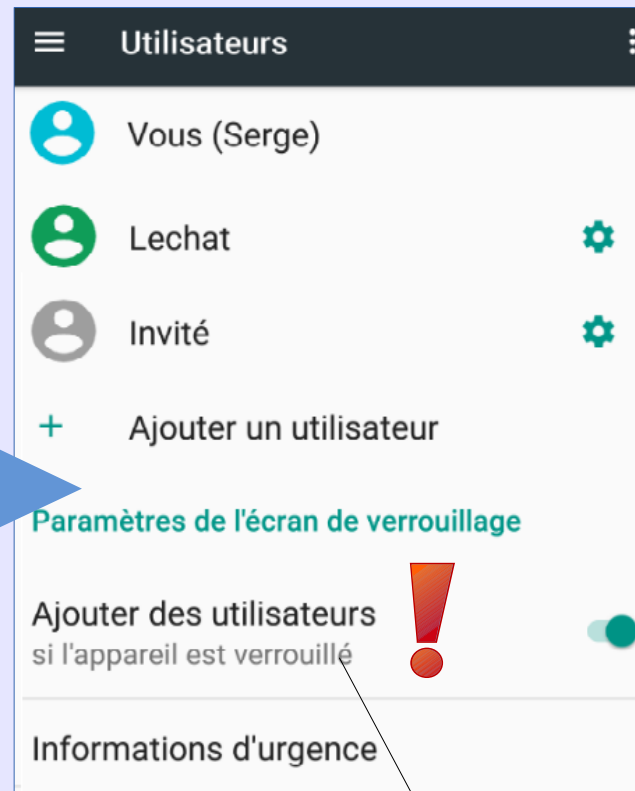
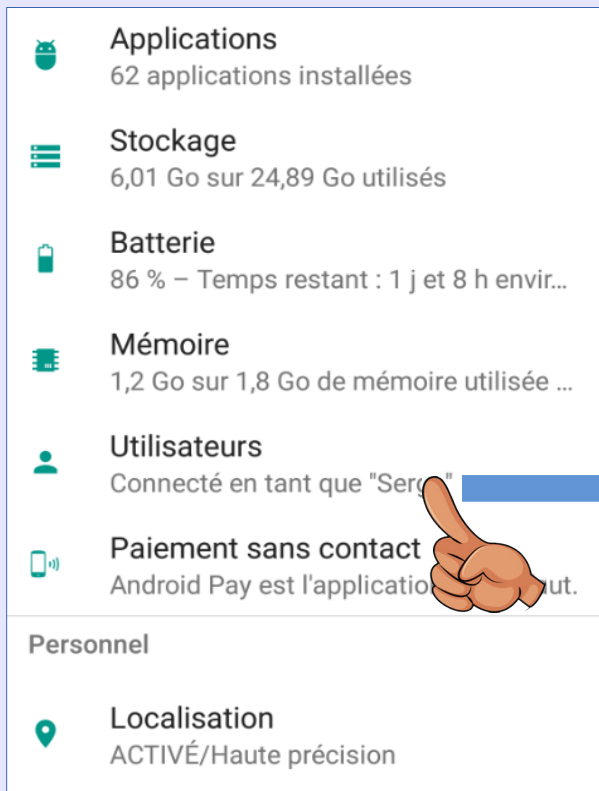
- Android 4.2 : introduction de la possibilité de créer plusieurs comptes utilisateurs sur un mobile
- Android 4.3 : introduction des profils restreints

## Intérêts

- Au départ : pouvoir partager un mobile (tablette) entre plusieurs personnes en préservant l'environnement de chacun ou pour restreindre les applications.
- Intéressant aussi pour séparer l'environnement professionnel du reste.

# Support multi-utilisateurs sous Android<sup>4.2 / 4.3 +</sup>

## Comptes utilisateurs



- Le compte *Lechat* a la possibilité de définir son propre compte Google
- Il voit les applications de base
- Il ne voit pas les applications installées par les autres utilisateurs.
- Il peut installer ses propres applications
- Il peut définir son propre code de verrouillage.
- Etc...

Permet de créer des utilisateurs ou invités sans déverrouiller

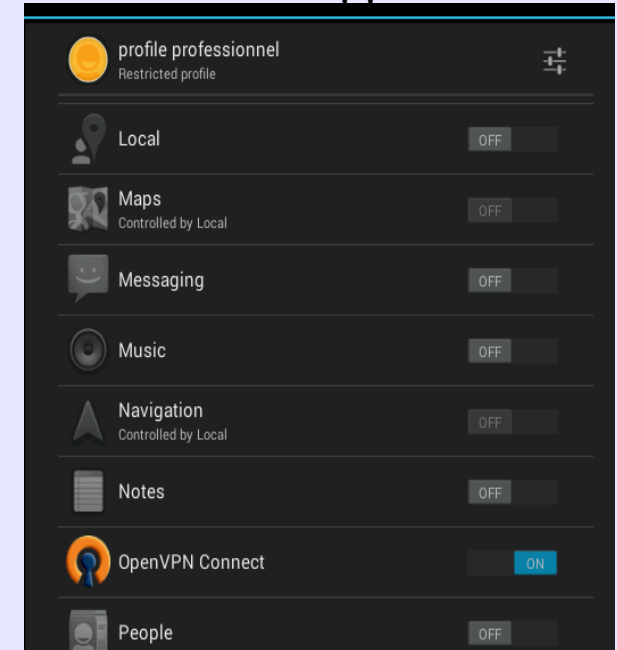
# Support multi-utilisateurs sous Android<sup>4.2 / 4.3 +</sup>

## Profiles restreints (uniquement sur tablettes)

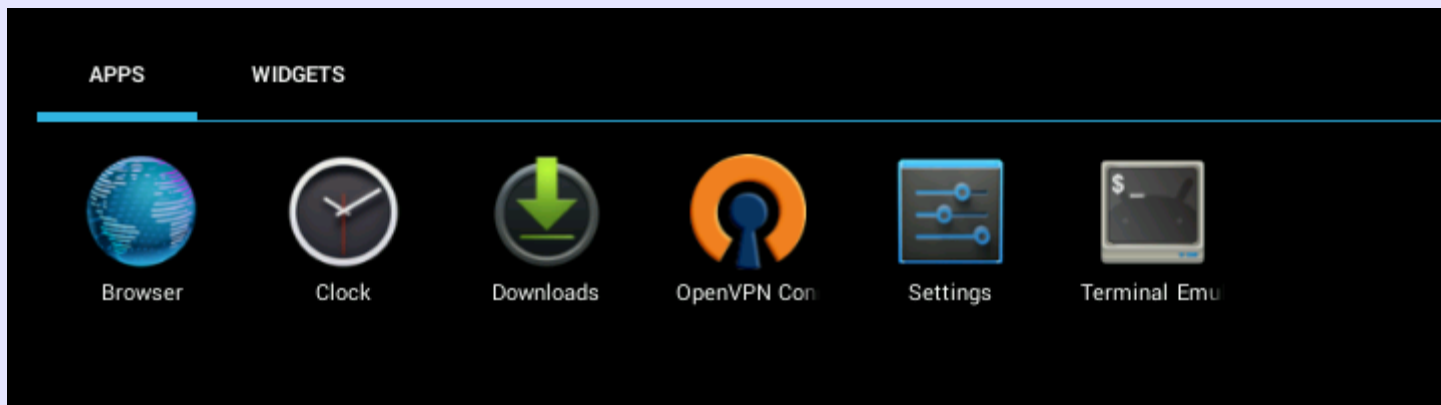
Mêmes caractéristiques que les comptes utilisateurs, avec en plus :

- Le compte du propriétaire du mobile doit obligatoirement avoir un code de verrouillage
- Seules les applications sélectionnées sont utilisables par le profile.

## Sélection des applications



## Applications utilisables depuis le profile



# Support multi-utilisateurs sous Android<sup>4.2 / 4.3 +</sup>

---

- Chaque utilisateur ou profile a un UID
  - L'utilisateur **owner** à l'UID 0
  - Les suivants : 10, 11, 12....

---

# Systeme de fichiers Android

---

# Système de fichiers Android<sup>4.3+</sup>

---

- Chaque application tourne avec son propre UID et GID
- Le kernel gère les droits (RWX) sur les fichiers
- Une application n'a pas accès aux fichiers d'une autre.
- C'est un système de type **Discretionary Access Control (DAC)** : Le propriétaire des données peut donner les droits qu'il veut à ses propres données.

# Système de fichiers Android<sup>4.3+</sup>

---

- La mémoire flash est divisée en 6 partitions :

- /boot
- /system
- /recovery
- /data
- /cache
- /misc



# Système de fichiers Android<sup>4.3+</sup>

---

- Chaque application a sa propre arborescence
  - Les exécutables dans **/system/app/<package-name>** (pré-installés) ou dans **/data/app/<package-name>**
  - Les data dans **/data/user/<uid>**
  - Chaque utilisateur a son propre répertoire data.
- Pour le owner dans **/data/user/0** alias **/data/data**
- Une seule instance des applications existe même s'il y a plusieurs utilisateurs.  
(cad : une application n'est pas installée une 2ème fois si elle a déjà été installée par un autre utilisateur.  
Mais le 2ème utilisateur suit le processus normal d'installation dans Google Play)

# Système de fichiers Android<sup>4.3+</sup>

## ● Contenu de /data/users

```
root@x86:/data/user # ls -l
lrwxrwxrwx root      root      2014-01-23 10:11 0 -> /data/data/
lrwxrwx--x system    system    2014-01-23 11:36 10
lrwxrwx--x system    system    2014-01-23 12:14 11
lrwxrwx--x system    system    2014-02-03 16:54 12
root@x86:/data/user #
```



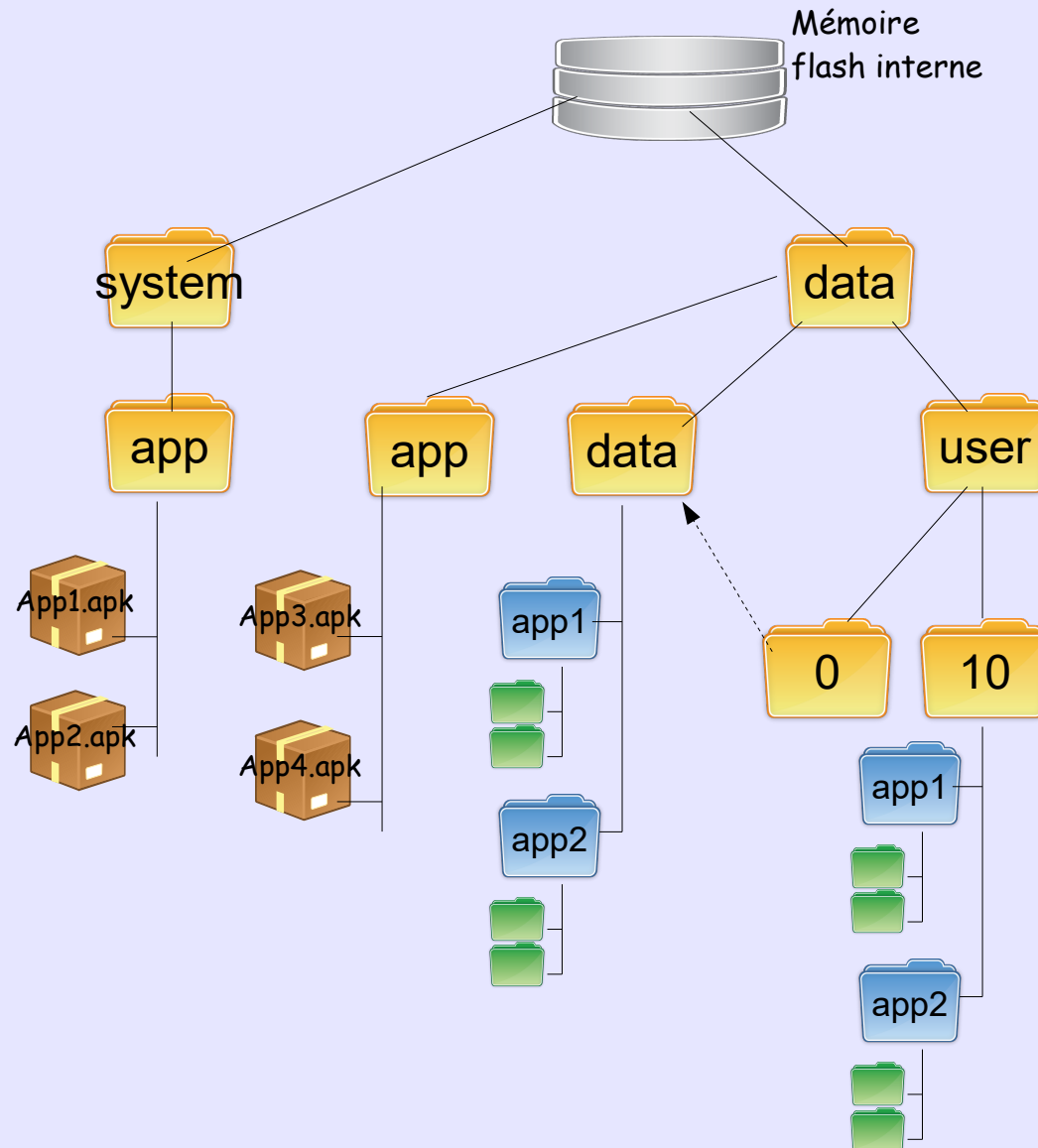
- Les UID 10 et 11 sont des utilisateurs
- L'UID 12 est un profile

## ● Contenu du répertoire /data/data alias /data/user/0

```
lrwxr-x--x u0_a16 u0_a16 2014-01-23 10:11 com.android.backupconfirm
lrwxr-x--x u0_a40 u0_a40 2014-01-23 10:11 com.android.basicmsreceiver
lrwxr-x--x bluetooth bluetooth 2014-01-23 10:11 com.android.bluetooth
lrwxr-x--x u0_a23 u0_a23 2014-01-23 10:14 com.android.browser
lrwxr-x--x u0_a27 u0_a27 2014-01-23 10:11 com.android.calculator2
lrwxr-x--x u0_a0 u0_a0 2014-01-23 10:11 com.android.calendar
lrwxr-x--x u0_a37 u0_a37 2014-01-23 10:15 com.android.certinstaller
lrwxr-x--x u0_a22 u0_a22 2014-01-23 10:11 com.android.contacts
lrwxr-x--x u0_a5 u0_a5 2014-01-23 11:02 com.android.defcontainer
lrwxr-x--x u0_a11 u0_a11 2014-01-23 10:11 com.android.deskclock
lrwxr-x--x u0_a47 u0_a47 2014-01-23 10:11 com.android.development
lrwxr-x--x u0_a52 u0_a52 2014-01-23 10:11 com.android.dialer
lrwxr-x--x u0_a44 u0_a44 2014-01-23 10:11 com.android.dreams.basic
lrwxr-x--x u0_a14 u0_a14 2014-01-23 10:11 com.android.email
lrwxr-x--x u0_a42 u0_a42 2014-01-23 10:11 com.android.exchange
lrwxr-x--x u0_a18 u0_a18 2014-01-23 10:11 com.android.galaxy4
lrwxr-x--x u0_a24 u0_a24 2014-01-23 10:11 com.android.gallery3d
```



# Système de fichiers Android<sup>4.3+</sup>

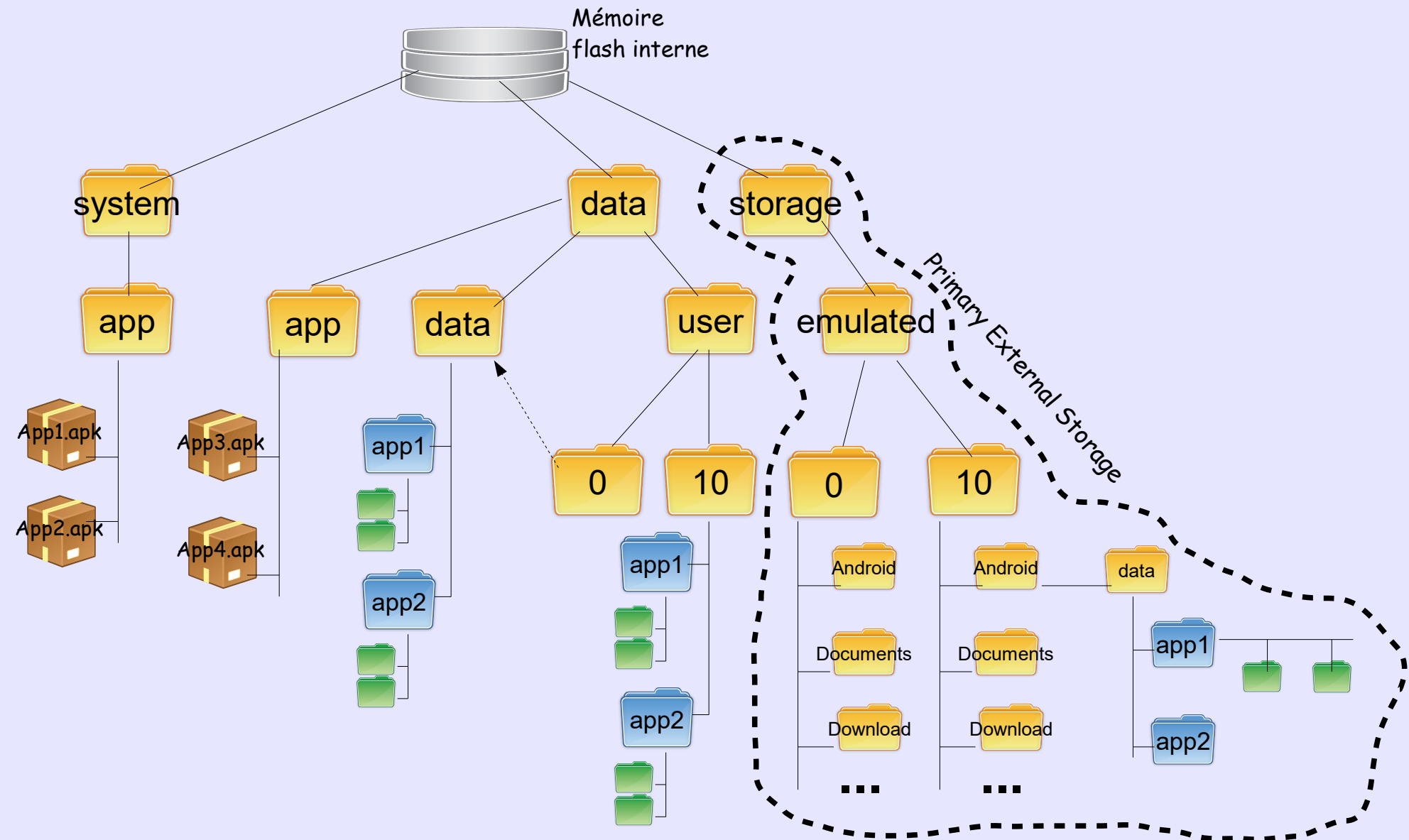


# 4.3+ Système de fichiers Android – Primary External Storage

---

- Dans les « anciennes » versions l'espace DATA était divisée en 2 partitions :
  - **/data** – pour les applications dans la mémoire flash interne.
  - **/mnt/sdcard** aussi appelée **sdcard** interne ou mémoire externe primaire !!! utilisée comme espace partagé
  - Problème : peu d'espace pour les applications, beaucoup pour l'espace partagé !
- Depuis 4.3 :
  - Il n'y a plus qu'une seule partition physique
  - La Sdcard est émulée grâce à **FUSE** (Filesystem in User Space) ou bien **sdcardfs** (wrapfs)
  - Cet espace émulé est appelé « **Primary external storage** »
  - L'ex sdcard est émulée dans **/storage/emulated/<uid>**


# 4.3+ Système de fichiers Android - Primary External Storage



# 4.3+ Système de fichiers Android - Primary External Storage

## ● Protection de l'espace partagé entre les utilisateurs


```
u0_a228@lt03wifi:/storage/emulated $ df
/mnt/secure/asec: Permission denied
/mnt/shell/container: Permission denied
/mnt/shell/emulated: Permission denied
Filesystem              Size      Used    Avail
/dev                    1.4G      140.0K    1.4G
/sys/fs/cgroup          1.4G      12.0K    1.4G
/mnt/secure             1.4G       0.0K    1.4G
/mnt/asec               1.4G       0.0K    1.4G
/mnt/asec/org.dmfs.caldav.lib-6 3.0M      1.4M    1.6M
/mnt/obb               1.4G       0.0K    1.4G
/system                2.3G      2.0G    322.1M
/efs                   19.7M      4.4M    15.3M
/cache                 196.8M      6.3M   190.5M
/preload               19.7M      8.1M    11.6M
/persdata/absolute     11.8M      4.1M     7.7M
/data                 11.8G      5.6G    6.1G
/data/data1            11.8G      5.6G    6.1G
/storage/emulated       1.4G       0.0K    1.4G
/mnt_1/sdcard_1        11.8G      5.6G    6.1G
/storage/emulated/0     11.8G      5.6G    6.1G
/storage/emulated/legacy 11.8G      5.6G    6.1G
1|u0_a228@lt03wifi:/storage/emulated $
```



- Un utilisateur ne voit que son propre espace partagé (primary external storage)
- Lorsqu'on active un utilisateur le système monte son espace partagé (`/storage/emulated/<uid>`) sur `/dev/media`

Exemple lorsque l'utilisateur owner est actif

```
1|u10_a33@x86:/ $ df
Filesystem              Size      Used    Free   Blksize
/                      1.1G      884.0K    1.1G   4096
/mnt                    1.1G      884.0K    1.1G   4096
/system                 1.1G      884.0K    1.1G   4096
/cache                  1.1G       0.0K    1.1G   4096
/data                  1.2G      633.9M   578.2M  4096
/dev                   1.1G      48.0K    1.1G   4096
/mnt/secure            1.1G       0.0K    1.1G   4096
/mnt/asec              1.1G       0.0K    1.1G   4096
/mnt/obb               1.1G       0.0K    1.1G   4096
/mnt/shell/emulated: Permission denied
/storage/emulated       1.1G       0.0K    1.1G   4096
/storage/emulated/10    1.2G      633.9M   578.2M  4096
/storage/emulated/10/Android/obb 1.2G      633.9M   578.2M  4096
/storage/emulated/legacy 1.2G      633.9M   578.2M  4096
/storage/emulated/legacy/Android/obb 1.2G      633.9M   578.2M  4096
1|u10_a33@x86:/ $
```



Exemple lorsque l'utilisateur uid=10 est actif

# 4.4+ Système de fichiers Android – Primary External Storage

---

- Protection de l'espace partagé vis à vis des applications


- Dans les anciennes versions d'Android toutes les applications avaient accès en lecture (permission **READ\_EXTERNAL\_STORAGE**) dans l'espace partagé et celles qui disposaient de la permission **WRITE\_EXTERNAL\_STORAGE** pouvaient y écrire aussi.
- Depuis **KitKat**, les choses ont changées :
  - Une application a tous les droits sur ses propres fichiers dans l'espace partagé.
  - Pour accéder aux fichiers des autres applications elle doit avoir reçu explicitement à l'installation un droit de lecture (**READ\_EXTERNAL\_STORAGE**) ou un droit d'écriture (**WRITE\_EXTERNAL\_STORAGE**)
  - Ces droits d'accès sont gérés à la fois par le système d'ACL classique d'Unix et par le driver FUSE.

# 4.4+ Système de fichiers Android - Primary External Storage

- Protection de l'espace partagé vis à vis des applications

- Lorsqu'on passe d'un utilisateur à l'autre, le système change le owner group du répertoire **/storage/emulated/<uid>** et le positionne à **sdcard\_r**.
- Le répertoire équivalent de l'ancien utilisateur courant est rendu à root.

```
u11_a228@lt03wifi:/ $ cd /storage/emulated
u11_a228@lt03wifi:/storage/emulated $ ls -l
d----- root      root      2014-12-15 11:44 0
drwxrwx--x root      sdcard_r 2014-12-12 15:33 11
drwxrwx--x root      sdcard_r 2014-12-12 15:33 legacy
u11_a228@lt03wifi:/storage/emulated $
```




- Les applications qui ont reçu la permission **READ\_EXTERNAL\_STORAGE** peuvent uniquement lire ce répertoire.



# 4.4+ Système de fichiers Android – Primary External Storage

- Protection de l'espace partagé vis à vis des applications

```
u11_a228@lt03wifi:/storage/emulated/11/Android/data $ ls -l
drwxrwx--- u11_a98  sdcard_r      2014-12-12 16:02 com.android.vending
drwxrwx--- u11_a41  sdcard_r      2014-12-12 15:32 com.dropbox.android
drwxrwx--- u11_a46  sdcard_r      2014-12-12 15:32 com.evernote
drwxrwx--- u11_a78  sdcard_r      2014-12-12 15:33 com.google.android.apps.magazines
drwxrwx--- u11_a55  sdcard_r      2014-12-12 15:33 com.google.android.apps.maps
drwxrwx--- u11_a58  sdcard_r      2014-12-12 15:56 com.google.android.gms
drwxrwx--- u11_a174 sdcard_r      2014-12-15 10:22 com.google.android.googlequicksearchbox
drwxrwx--- u11_a88  sdcard_r      2014-12-12 15:33 com.google.android.music
drwxrwx--- u11_a176 sdcard_r      2014-12-12 15:33 com.google.android.videos
drwxrwx--- u11_a182 sdcard_r      2014-12-12 15:33 com.google.android.youtube
drwxrwx--- u11_a222 sdcard_r      2014-12-15 14:27 com.owncloud.android
drwxrwx--- u11_a253 sdcard_r      2014-12-12 16:05 com.peel.app
drwxrwx--- u11_a116 sdcard_r      2014-12-12 15:32 com.samsung.android.snote
drwxrwx--- root    sdcard_r      2014-12-12 16:01 com.sec.android.allshare
drwxrwx--- u11_a136 sdcard_r      2014-12-12 15:33 com.sec.android.gallery3d
drwxrwx--- u11_a10  sdcard_r      2014-12-12 16:01 com.sec.android.nearby.mediaserver
drwxrwx--- u11_a51  sdcard_r      2014-12-12 15:33 flipboard.app
drwxrwx--- u11_a77  sdcard_r      2014-12-12 15:33 sstream.app
```




→ Userid de chaque application

# 4.3+ Système de fichiers Android - Primary External Storage


- Résultat de la commande **mount** (avec Android X86 , 4.3)

```
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
proc /proc proc rw,relatime 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/secure tmpfs rw,relatime,mode=700 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
/dev/fuse /mnt/shell/emulated fuse rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,default_permissions,allow_o
ther 0 0
tmpfs /storage/emulated tmpfs rw,nosuid,nodev,relatime,mode=050,gid=1028 0 0
/dev/fuse /storage/emulated/0 fuse rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,default_permissions,allow_o
ther 0 0
/dev/fuse /storage/emulated/0/Android/obb fuse rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,default_permiss
ions,allow_other 0 0
/dev/fuse /storage/emulated/legacy fuse rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,default_permissions,al
low_other 0 0
/dev/fuse /storage/emulated/legacy/Android/obb fuse rw,nosuid,nodev,relatime,user_id=1023,group_id=1023,default_pe
rmissions,allow_other 0 0
u0_a33@x86:/storage $
```



- Résultat de la commande **mount** (sur Galaxy note 10.1 2014)

```
ign 0 0
/data/media /storage/emulated/0 sdcardfs rw,nosuid,nodev,relatime,uid=1023,gid=1023
) 0
/data/media /storage/emulated/0/Android/obb sdcardfs rw,nosuid,nodev,relatime,uid=10
```

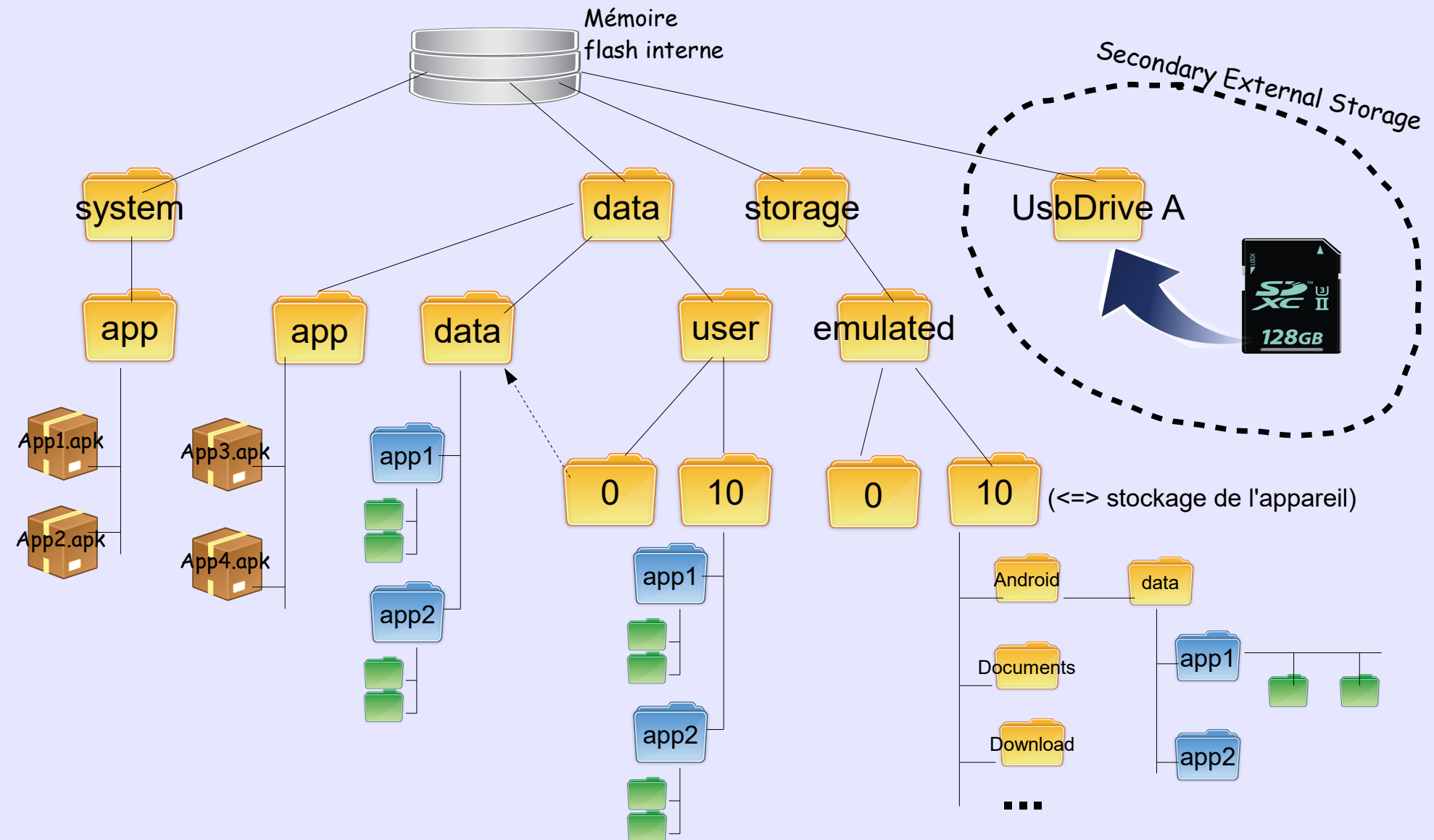


# Système de fichiers Android<sup>4.3+</sup> - Secondary External Storage

---

- Il est possible d'étendre le système de fichiers (**Secondary external storage**)
  - En connectant une carte SD externe ou une clé USB (stockage amovible)
  - Ce volume est monté sur **/storage/UsbDriveA**

# 4.3+ Système de fichiers Android - Secondary External Storage



# Système de fichiers Android - Secondary External Storage

---

- Avant la version 4.4 les applications pouvaient lire et écrire n'importe où dans la **secondary external storage**
- Depuis la version 4.4 :
  - chaque application peut écrire uniquement dans son propre espace.
  - Toutes les applications peuvent tout lire

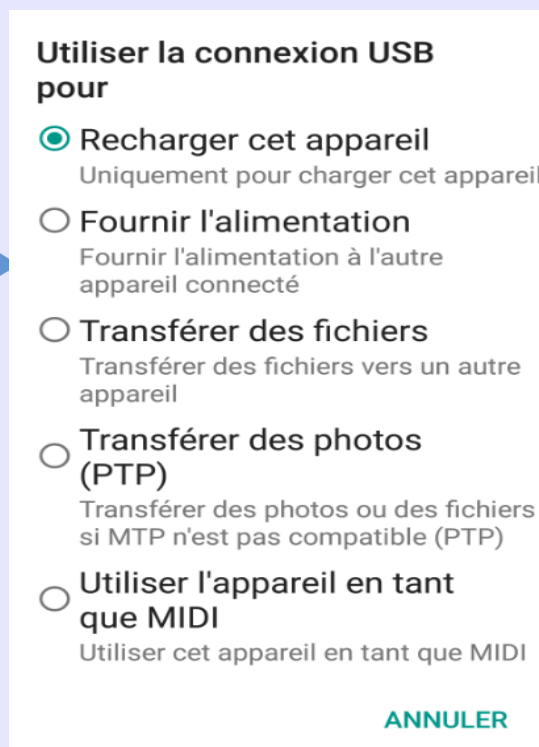
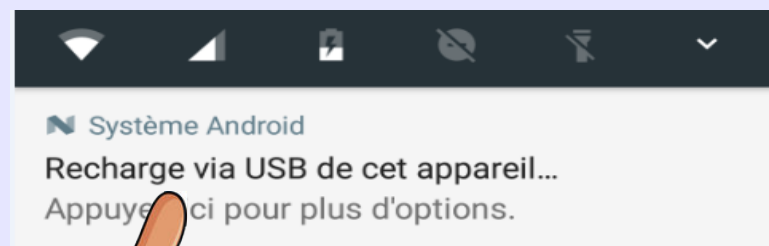
# Système de fichiers Android : Connexion USB

---

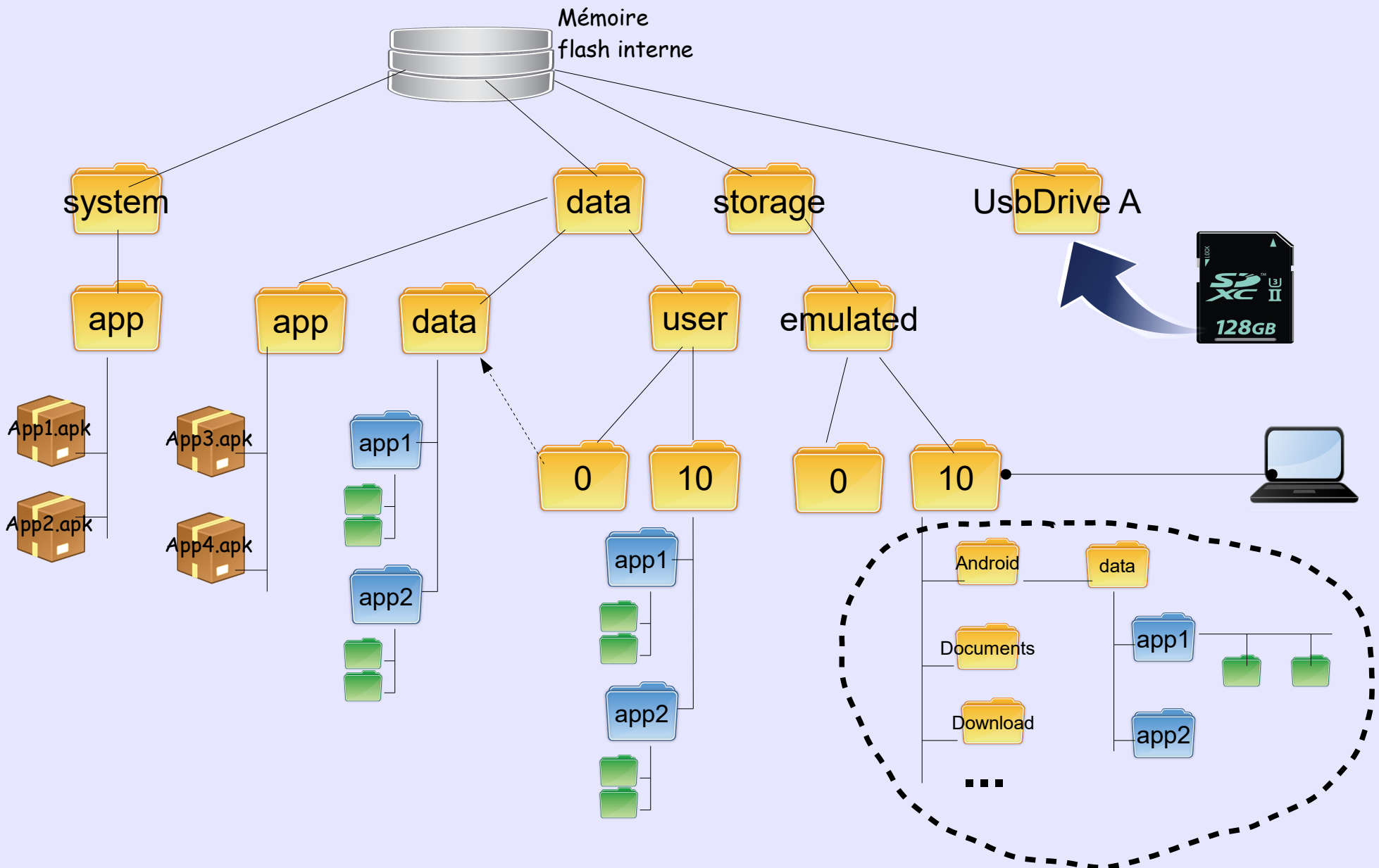
- Il est possible de relier un système Android à un ordinateur par USB
- L'écran doit être déverrouillé pour que l'ordinateur puisse monter le volume.
- Seul le primary external storage de l'utilisateur courant est visible depuis l'ordinateur
- Tout type de fichiers peut y être copié et sera visible par toutes les applis autorisées à utiliser le primary external storage

# Système de fichiers Android : Connexion USB

- Depuis Lollipop, le simple fait de brancher un câble USB ne suffit plus pour accéder aux fichiers
- Par défaut le mobile se met en charge, il faut explicitement lui permettre de faire autre chose



# 4.3+ Système de fichiers Android : Connexion USB





---

## Chiffrement des mobiles Android

---



# Chiffrement sous Android

---

- Support (natif) du chiffrement **full-disk** à partir de la version 5.0
  - x Une seule clé protège toute la partition DATA (dmccrypt)
  - x la clé est protégé par le code déverrouillage utilisateur
  - x Avant 5.0, ce chiffrement devait être activé volontairement par l'utilisateur.
  - x Le chiffrement est complètement transparent pour toutes les applications.  
Tous les fichiers créés par les applications sont donc protégés.
- Support du chiffrement **file-based** à partir de la version 7.0
  - x Chaque fichier peut être chiffré avec une clé différente
  - x Chaque fichier peut-être déchiffré indépendamment des autres.

# Chiffrement sous Android : full-disk encryption (FDE)

---

L'environnement de cryptographie sous Lollipop a été renforcé pour durcir la résistance aux attaques offline.

- La partition **/data** est chiffrée lors du premier boot
- Seuls les blocs déjà utilisés sont chiffrés (fast encryption) pour accélérer le 1<sup>er</sup> boot
- La master key (Disk Encryption Key) générée est initialement chiffrée avec un mot de passe par défaut (=default\_password)
- Lorsque l'utilisateur positionnera son propre passcode cette master key sera re-chiffrée (les data ne sont pas rechiffrées)
- Possibilité d'utiliser un modèle, un pin code ou un mot de passe alphanumérique.
- Lollipop peut utiliser la technologie **ARM TrustedZone** du processeur (**hardware-backed storage**) rendant impossible les attaques offline sur la clé.

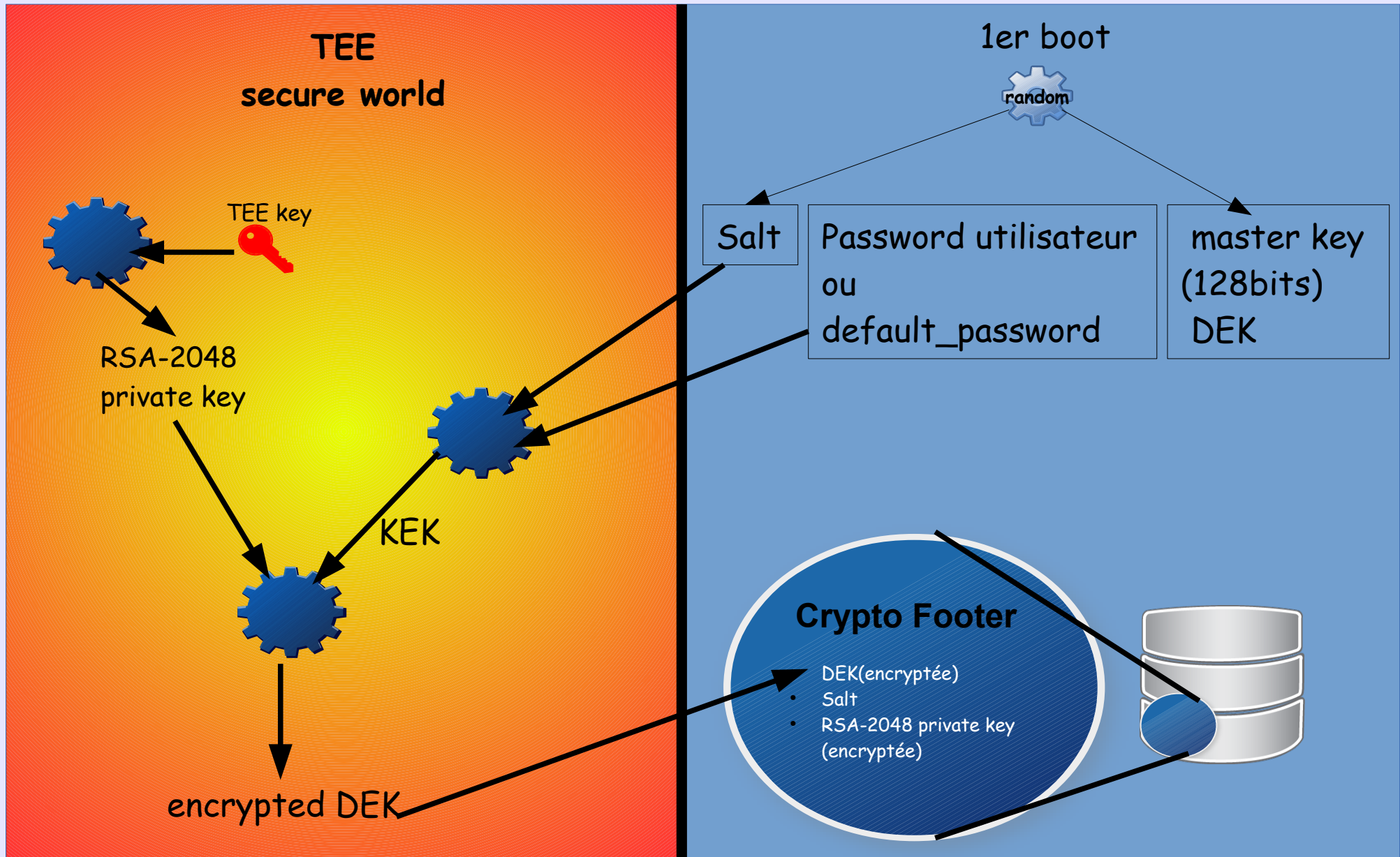
# Chiffrement sous Android : full-disk encryption (FDE)

---

## ARM trustedZone

- Technologie embarquée dans le processeur lui permettant de tourner dans deux environnements :
  - Normal world : c'est là que s'exécute le système d'exploitation.
  - Secure world : accessible uniquement à travers les fonctionnalités du **TEE** (Trusted Execution Environment)


# Chiffrement sous Android : full-disk encryption (FDE)



# Chiffrement sous Android : full-disk encryption (FDE)


Montage avant chiffrement

```
root@android:/ # mount
rootfs / rootfs ro,relatime 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
/dev/block/sda6 /system ext4 ro,relatime,data=ordered 0 0
/dev/block/sdb1 /cache ext4 rw,nosuid,nodev,relatime,data=ordered 0 0
/dev/block/sdb3 /data ext4 rw,nosuid,nodev,relatime,data=ordered 0 0
/dev/block/sdc /mnt/sdcard vfat rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,cod
root@android:/ #
```



Montage après chiffrement

```
rootfs / rootfs ro,relatime 0 0
tmpfs /dev tmpfs rw,nosuid,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
/dev/block/sda6 /system ext4 ro,relatime,data=ordered 0 0
/dev/block/sdb1 /cache ext4 rw,nosuid,nodev,relatime,data=ordered 0 0
none /mnt/shared/android-extension vboxsf rw,nodev,relatime 0 0
/dev/block/sdc /mnt/sdcard vfat rw,relatime,fmask=0000,dmask=0000,allow_utime=0022,cod
/dev/block/dm-0 /data ext4 rw,nosuid,nodev,relatime,data=ordered 0 0
root@android:/ #
```



# Chiffrement sous Android : file-based encryption (FBE)

---

- Disponible à partir de 7.0
- Sans FBE, l'utilisateur doit donner son code de déverrouillage au boot pour permettre l'exécution des applis.
- Avec FBE, des applis peuvent être démarrées avant le déverrouillage (**Direct Boot**)

L'espace de stockage est divisé en 2 parties

- **Credential encrypted** (CE) : Correspond à l'espace protégé par FDE et disponible qu'après le 1<sup>er</sup> déverrouillage de l'appareil
- **Device encrypted** (DE) : espace disponible pendant le Direct Boot et avant le déverrouillage

---

## Permissions sous Android

---



# Permissions sous Android

---

- Les permissions permettent de gérer l'accès aux ressources du système
- C'est une autre partie du sandboxing Android
- Chaque application doit déclarer explicitement les permissions dont elle a besoin (manifest)
- Il existe une centaine de permissions
- Lors de l'installation, l'utilisateur doit accepter, ou pas, de délivrer les permissions à l'application
- Obligation d'accepter tout ou rien

# Permissions sous Android

---

- Les permissions d'une application peuvent être vues :
  - Au moment de l'installation (si installée depuis Google Play)
  - Dans les paramètres du mobile, dans la gestion des applications.
- Pour le moment pas de possibilités de gérer ses permissions après installation.

# SE For Android

---

- SE (Security Enhancement) for Android est une implémentation de SELINUX
- Introduit à partir de 4.3 en mode permissive et en mode Enforcing en 4.4
- Le but est de limiter les conséquences d'une vulnérabilité qui permettrait à du code malveillant de s'exécuter dans un processus système qui tourne sous root.
- En fait, pour réduire la puissance du compte root, qui n'a plus tout les droits.
- Mandatory Access Control (MAC):
  - Les possibilités d'un processus sont limitées par une politique système globale (par exemple, même un processus qui tourne sous root n'aura pas accès à tout le système)
- Renforce l'isolation des processus.
- Centralise la politique de sécurité

---

# Gestion des comptes pour les applications

---



# Account Manager Service

---

- Dès qu'une application installée offre un service client vers un serveur distant des identifiants sont nécessaires.
- La plupart des applications enregistrent les identifiants
- La question est : comment sont enregistrés ces identifiants ?
- Soit l'application gère cela dans son propre espace data, et on ne sait pas ce qu'elle fait, soit elle utilise l' **AccountManagerService**.

# Account Manager Service

---

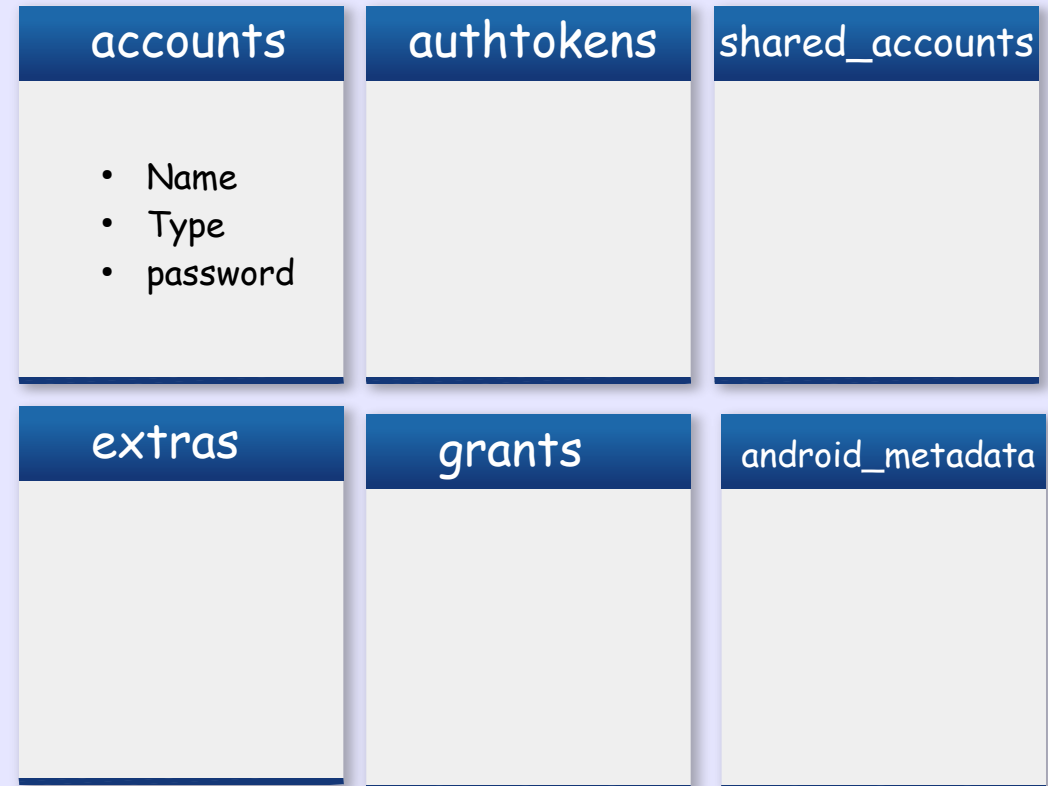
- L' **Account Manager Service** gère une base centrale des comptes par utilisateur du mobile
- Cette table se trouve dans **/data/system/users/<userid>/account.db**
- Elle n'est accessible pour les applications qu'à travers l'API de l'Account Manager Service
- Il faut accorder des permissions aux applications pour qu'elles puissent utiliser l'API
- L'account Manager Service permet de gérer des mots de passe ou des **tokens**

# Account Manager Service : account.db

- **account.db** contient 6 tables

- La table **accounts** contient :

- Name : l'identifiant (login)
- Type : contient le nom de l'application
- Password : contient le password



- L'Account Manager ne gère pas le chiffrement du mot de passe. C'est de la responsabilité de l'application.
- Android X86 permet de voir très facilement si une appli chiffre le mot de passe. En général non.

# Account Manager Service : account.db

---

- Interrogation de la base **accounts.db**

```
cd /data/system/users/0  
sqlite3 accounts.db  
select * from accounts ;
```



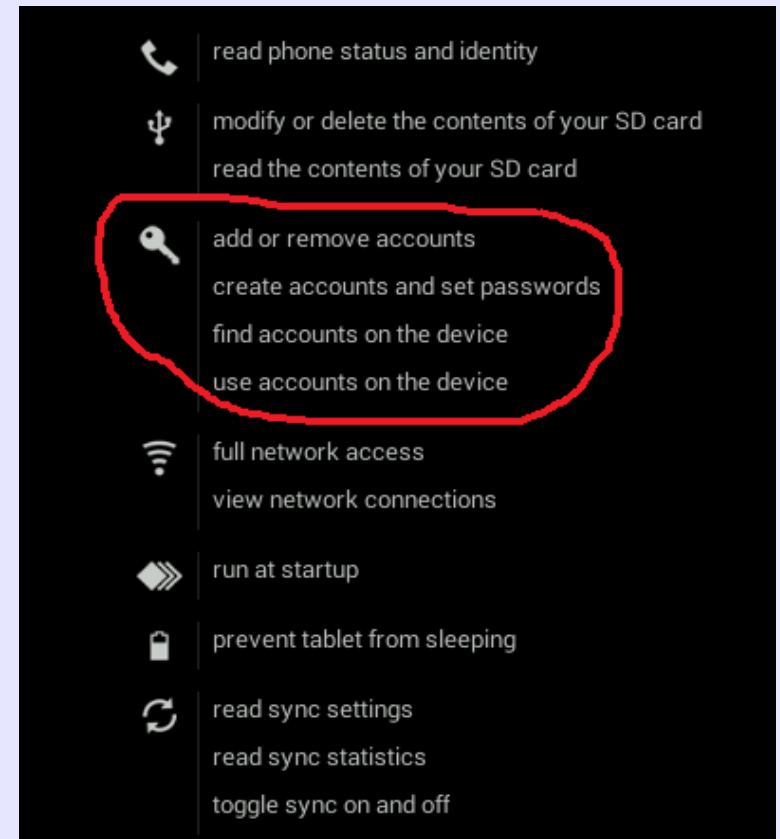
```
Pierre.dupont@gmail.com | comm.google | oauth2rt_1/sskjssjj9KJSnksdhk877çkçaa  
dupont@labo.cnrs.fr | com.android.email | motdepasseenclair ←  
dupont@owncloud.labo.cnrs.fr | owncloud | motdepasseenclair ←
```

- On voit ici que les applications mail d'Android et owncloud stockent le mot de passe **en clair**.



# Account Manager Service : account.db

- La liste des comptes créés (et donc les applications qui utilisent Account Manager) peut être consultée dans **paramètres/Comptes**.
- Si une application n'est pas dans cette liste c'est qu'elle gère les identifiants autrement.
- Les permissions des applications utilisant Account Manager peuvent être consultées dans **paramètres/gestionnaire d'applications**



# Account Manager Service : authentication Google

- Tout utilisateur d'un mobile Android doit disposer d'un compte Google (et donc d'un mot de passe) pour pouvoir accéder à Google Play.
- Lors de la première authentification depuis son mobile, l'utilisateur donne son identifiant et mot de passe Google. Ce Mot de passe **n'est pas** enregistré dans le mobile.
- Le serveur d'authentification Google est contacté et renvoi un **master token** qui est stocké à la place du mot de passe dans **/data/system/users/0/accounts.db**
- C'est ce token qui est utilisé pour les connexions suivantes sur les services Google.

```
cd /data/system/users/0  
sqlite3 accounts.db  
select * from accounts ;
```



```
Pierre.dupont@gmail.com | comm.google | oauth2rt_1/sskjssjj9KJSnksdhk877çkçaa  
dupont@labo.cnrs.fr | com.android.email | motdepasseenclair  
dupont@owncloud.labo.cnrs.fr | owncloud | motdepasseenclair
```



# Account Manager Service : authentication Google

---

- Ce token ne peut servir qu'à partir du mobile pour les services Google et pas pour se connecter sur le compte Google avec un navigateur.
- Le token est généré par l'utilisation du protocole OAuth 2.0 (utilisé aussi sur Facebook, Twitter)

# III

---

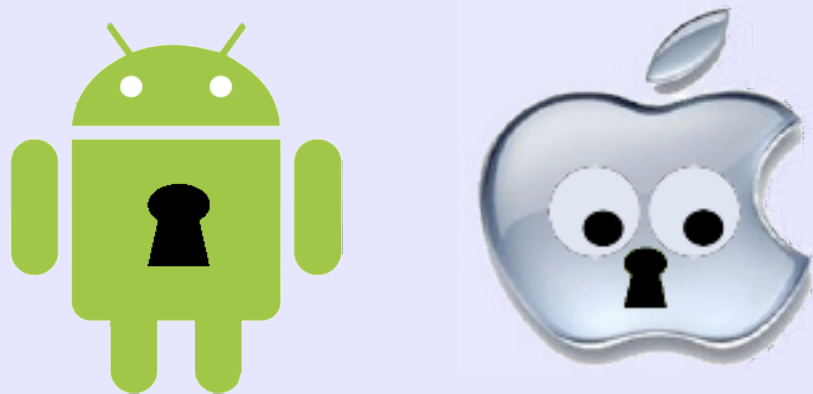
## Fonctionnalités

---

---

# Verrouillage du mobile

---



# Verrouillage du mobile

---

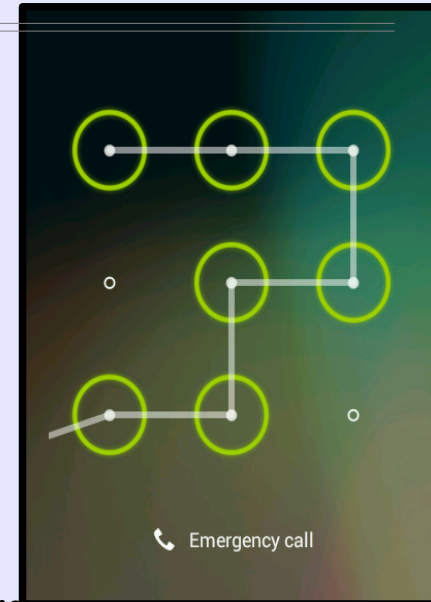
- Le verrouillage d'écran est la protection **minimale et obligatoire**  
Tout mobile en relation avec le S.I doit verrouiller son écran
- Rien à voir avec le pincode de la carte SIM qui protège uniquement l'utilisation de cette carte

# Verrouillage du mobile : Par modèle



## Consiste à dessiner un motif entre les points contigus

- La robustesse est proportionnelle à la complexité du motif
- Réputé peu robuste car le nombre de combinaisons est faible :
  - 4 points => 1624 solutions
  - 5 points => 7152 combinaisons
  - 6 points => 26016 combinaisons ==> plus de combinaisons qu'un pincode à 4 chiffres
  - 9 points => 140704 combinaisons
- Attention aux traces de doigts (smudge attack)



0	1	2
3	4	5
6	7	8

# Verrouillage du mobile : Par pincode



## Verrouillage par une suite de 4 à 16 chiffres

- Robustesse proportionnelle au nombre de chiffres.
- En général 4 chiffres utilisés => 10000 combinaisons seulement
- Attaque force brute par root ou jailbreak, pas directement sur l'écran



# Verrouillage du mobile : par mot de passe alpha-numérique



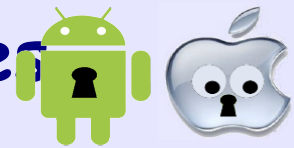
## Utilisation d'un « vrai » mot de passe alpha-numérique

- Méthode la plus robuste
- Mais la moins pratique
- Difficile de taper un mot de passe contenant chiffres, lettres, caractères spéciaux sur un smartphone :
  - basculement entre les claviers virtuels (numérique, alpha)
  - gros doigts - petites touches
  - Soleil...

Un intérêt important d'un smartphone est sa capacité multifonction qui nécessite de pouvoir le « dégainer » rapidement (téléphone, photo, gps, accès Internet, ou autre application).

Un mot de passe alpha-numérique est un gros frein qui fait perdre de l'intérêt au smartphone qui découragera les utilisateurs.

# Verrouillage du mobile : Nouvelles méthodes



## Empreintes digitales

- Apparu récemment sur les iPhone 5 d'Apple (TouchId)
- Présent désormais sur matériels Android



# Déverrouillage du mobile : Nouvelles méthodes

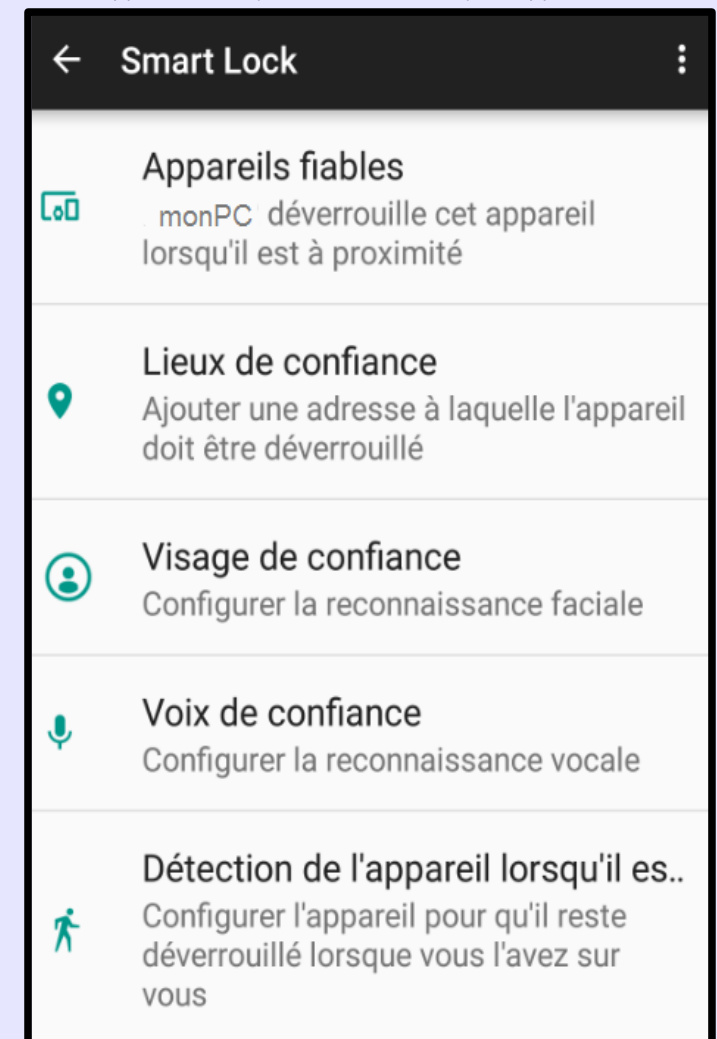


## Smart lock Android

- Depuis la version 5.0, **Smart lock** permet de déverrouiller automatiquement l'écran lorsque l'appareil est dans un environnement de confiance
  - En présence d'un appareil reconnu (bluetooth ou NFC)
  - Sur un lieu géographique prédéfini (GPS)
  - Par reconnaissance d'un visage ou de voix...

(Peut servir pour se procurer un moyen de déverrouillage de secours)

## Paramètres/sécurité/smart lock





---

## Windows Unlock

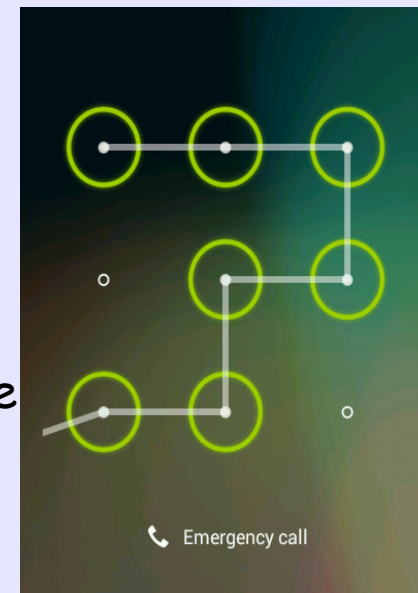
- Possibilité de déverrouillage un PC Windows 10 avec un objet compagnon, un smart phone par exemple.
- Pas encore disponible

# Verrouillage du mobile : stockage du passcode



## Verrouillage par modèle

- Ici le modèle vaut : 00010204050607
- Il est stocké dans le fichier **/data/system/gesture.key** sous forme d'un hash SHA-1 sans salt. Ce fichier n'est pas accessible pour l'utilisateur.
- En cas de dump de la partition le hash est facilement déchiffrable.
- La destruction du fichier supprime le passcode



# Verrouillage du mobile : stockage du passcode



## Verrouillage par PIN code et mot de passe

- Le PIN code ou le mot de passe sont stockés dans le fichier **/data/system/users/<userid>/password.key**  
ou  
**/data/system/password.key** pour le owner
- Hash SHA-1 + MD5 avec un salt de 64 bit
- Le salt et type de passcode sont stockés dans une base sqlite :  
**/data/system/locksettings.db** (pour tous les utilisateurs)

```
3lroot@x86:/data/system #  
3lroot@x86:/data/system # sqlite3 locksettings.db  
SQLite version 3.7.11 2012-03-20 11:35:50  
Enter ".help" for instructions  
Enter SQL statements terminated with a ";"  
sqlite> select * from locksettings where user=10;  
15|lockscreen.password_salt|10|-7430867514089410237  
16|lock_pattern_autolock|10|0  
18|lockscreen.password_type_alternate|10|0  
19|lockscreen.password_type|10|131072  
20|lockscreen.passwordhistory|10|  
sqlite>
```



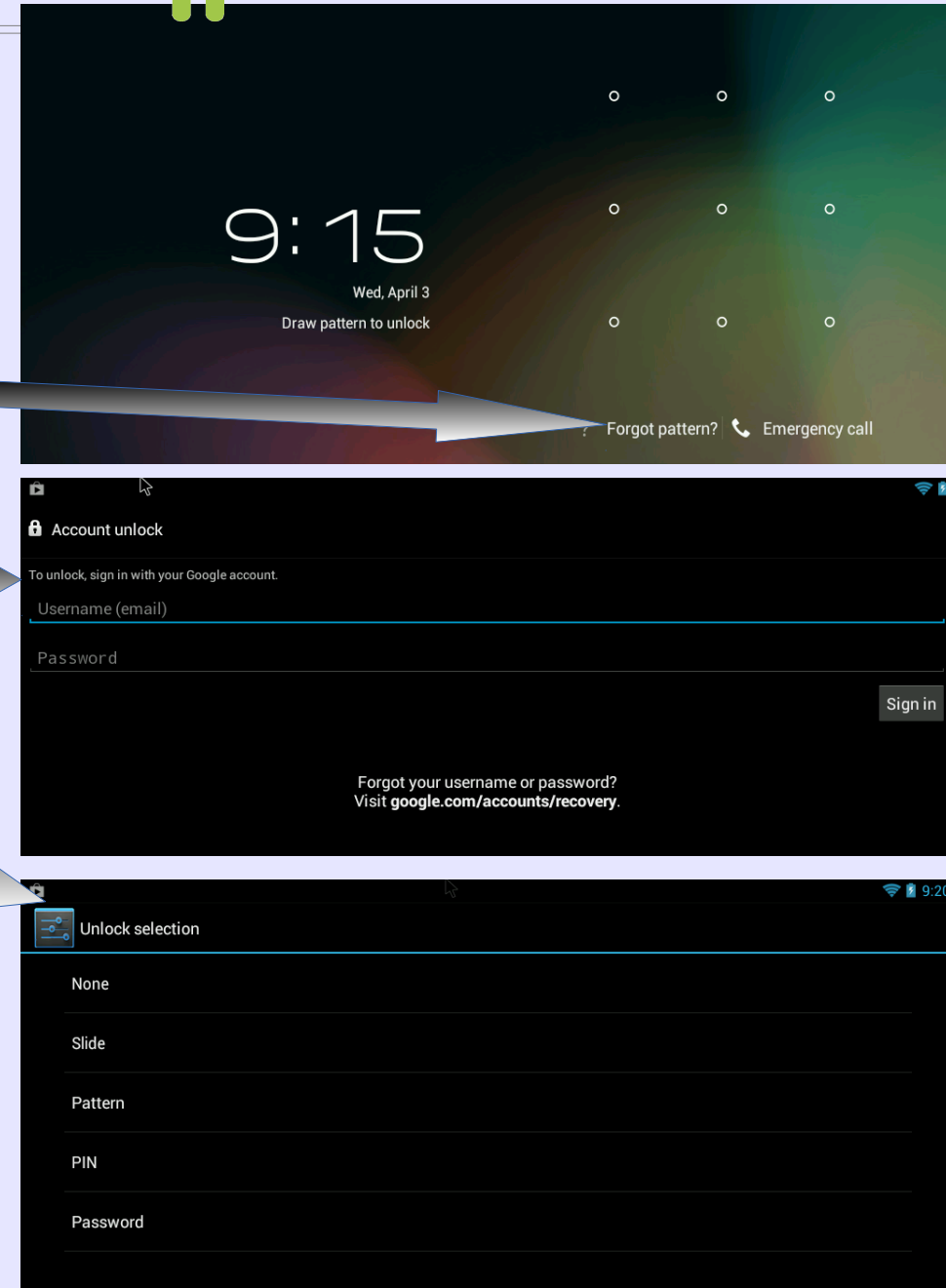
# Perte du code de verrouillage



## Méthode « locale »

- Un mobile Android se bloque pendant 30 secondes à la cinquième tentative de déverrouillage avec un code erroné. (et ainsi de suite toutes les 5 tentatives)
- Il est possible alors de sélectionner l'option « Mot de passe oublié »
- Le login et le mot de passe du compte Google sont alors demandé.
- Une fenêtre propose de choisir un nouveau mode de verrouillage.

**Cela marche uniquement si le réseau est activé et connecté**



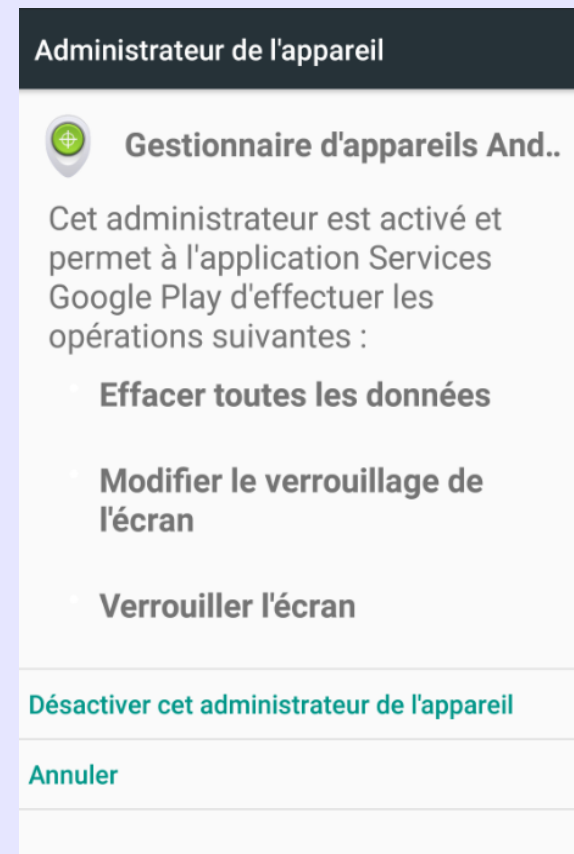
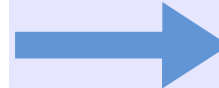
# Perte du code de verrouillage



## Méthode « à distance »

- Il faut **d'abord** activer le « Gestionnaire d'appareil Android » **avant d'avoir un problème**

Paramètres/sécurité/Admin. de périphérique



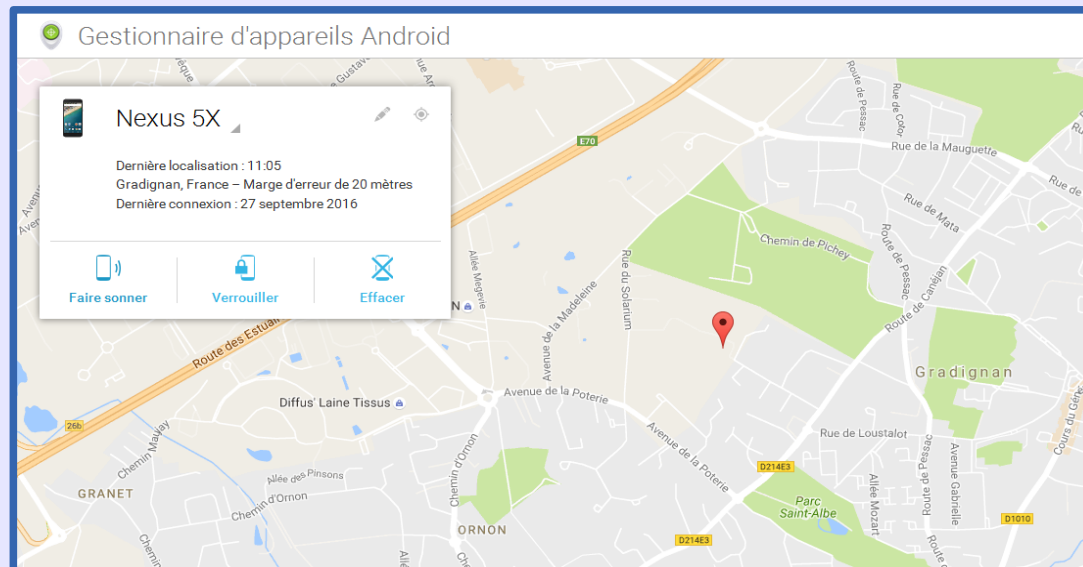


# Perte du code de verrouillage



## Méthode « à distance »

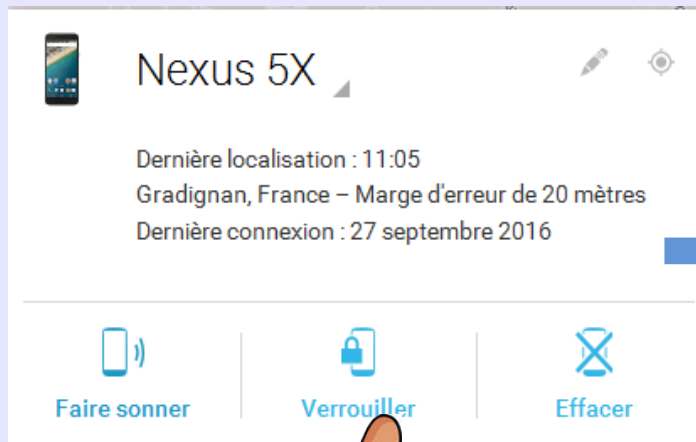
- Pour ré-positionner un mot de passe d'écran  
Se rendre : <https://www.google.com/android/devicemanager>  
(mot de passe Google nécessaire)
- Certains constructeurs peuvent remplacer le service Google par leur propre service.  
Ex : Samsung : <http://findmymobile.samsung.com>
- Le mobile doit être sur le réseau



# Perte du code de verrouillage



## Méthode « à distance »



Nouvel écran de verrouillage

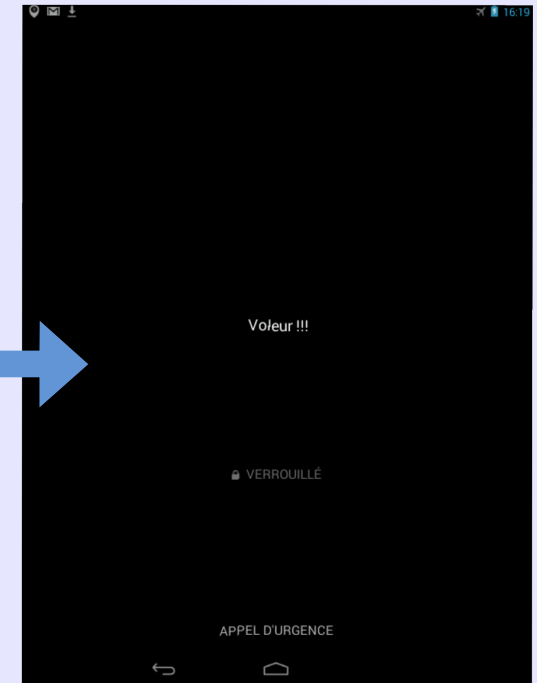
Votre écran de verrouillage actuel sera remplacé par un verrouillage par mot de passe. N'utilisez pas le mot de passe de votre compte Google.

Nouveau mot de passe  
.....

Confirmer le mot de passe  
.....

Message de récupération (facultatif)  
Ce message s'affichera sur votre écran de verrouillage

Annuler Verrouiller



# Perte du code de verrouillage



## Si le mobile a été sauvegardé dans iTunes

- Il faut le restaurer => un nouveau passcode est demandé.

## Si le mobile n'a pas été sauvegardé

- Le mobile devra être réinitialisé, les données seront effacées

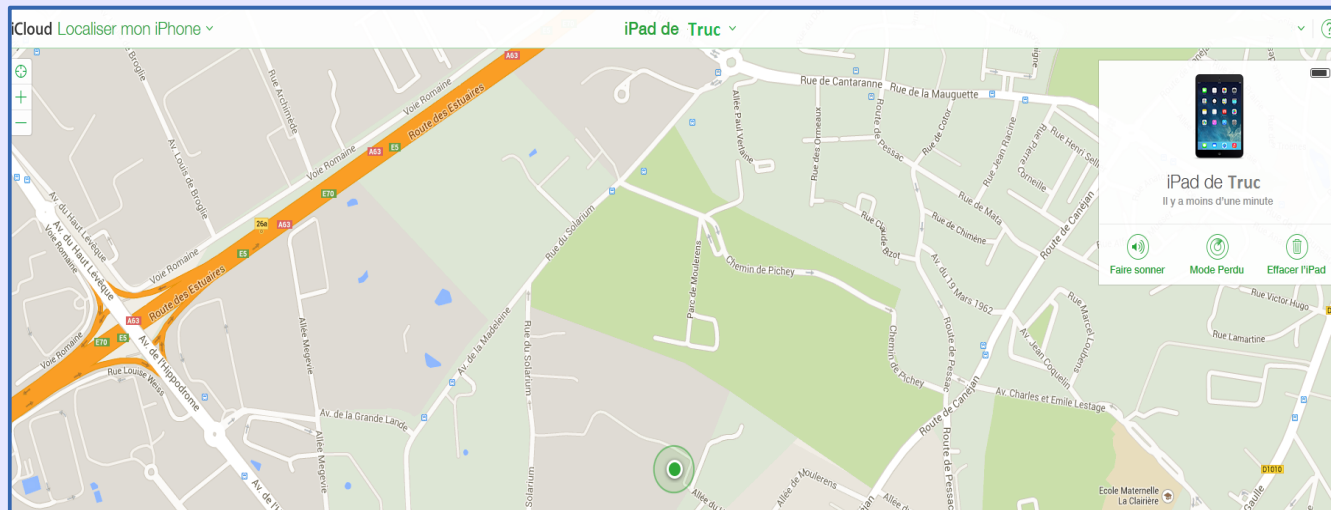
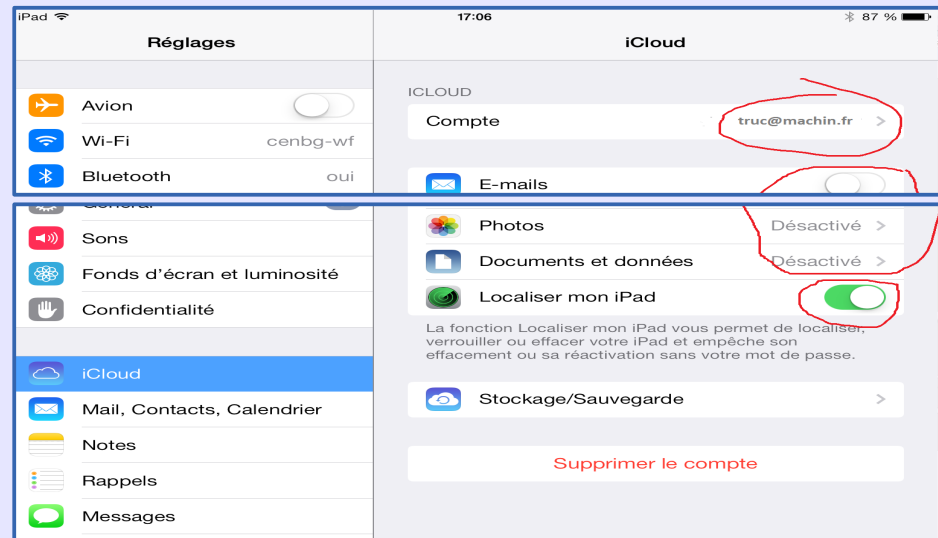
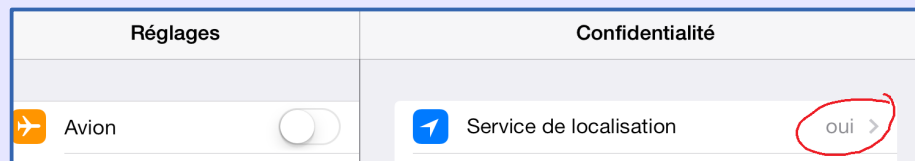
[http://support.apple.com/kb/HT1212?viewlocale=fr\\_FR](http://support.apple.com/kb/HT1212?viewlocale=fr_FR)

# Contrôle à distance



## Avant le problème

- Il faut créer un compte iCloud et activer la fonction « Localiser mon iPhone/iPad »
- Activer le service de localisation



# Effacement des données

---



## Mobile non chiffré, avant 4.3

- Factory reset détruit la partition **/data** => Les données ne sont pas réellement effacées
- Chiffrer le mobile , puis factory reset



## Mobile non chiffré >= 4.3

- Le factory reset est suivi d'une commande TRIM vers la NAND qui efface les blocs de données
- Pour être sûr => Chiffrer puis factory reset.



## Mobile chiffré toutes versions

- Factory reset

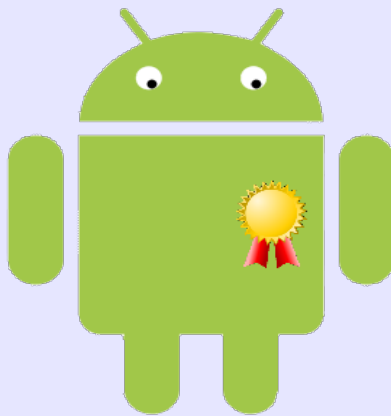


- Fonction Wipe : détruit la clé de chiffrement

---

## Certificats sous Android

---



# Les certificats sur un mobile (en général)

---

## Quels intérêts ?

- Chaque mobile peut être identifié/authentifié par un certificat
- L'importation d'un certificat **force l'usage d'un code de verrouillage** du mobile
- Un certificat a une durée de vie limitée (un certificat compromis finit par être inopérant)
- En cas de compromission du mobile/certificat, il suffit de révoquer le certificat, seul le mobile est impacté.
- Permet l'accès à un réseau Wifi sécurisé (EAP-TLS)
- Permet l'accès distant sécurisé via VPN

# Les certificats sur un mobile (en général)

---

## Structure nécessaire

- Il faut disposer d'une **IGC** pour créer les certificats
- Il faut mettre en place une procédure pour que le certificat arrive jusqu'au mobile
- Pour l'autorisation d'accès, un serveur RADIUS

## Quels certificats ?

- Techniquement les certificats CNRS sont utilisables
- Comme il s'agit d'une utilisation purement locale ce n'est pas très souhaitable (procédure de création, renouvellement, diffusion très lourde et peu adaptée)
- La création d'une IGC locale peut rendre le processus très simple (et bénéficier à l'ensemble du parc).



# Les certificats sous Android

---

- Android dispose d'un magasin de certificats (Keychain)  
A la différence d'IOS, Android ne stocke que des certificats dans le keychain.
- La gestion se fait au travers du menu « Paramètres/sécurité » puis le paragraphe « **Stockage des informations d'identification** »
- L'importation se fait à partir de fichiers PKCS12 (P12) ou CRT pour les certificats des autorités. (fichiers détruits après l'importation).
- L'enregistrement du premier certificat **exigera** que le mobile soit protégé par un moyen de verrouillage (modèle, pin code ou mot de passe).
- Il n'y a pas de fonction d'exportation de la clé privée du certificat
- La gestion des certificats a été revue lors du passage en 4.3
  - Support du « **hardware-backed** »
  - Séparation certificats WIFI - certificats VPN et applications
  - Support du multi-utilisateurs : chaque utilisateur dispose de son propre magasin de certificat.

# Les certificats sous Android

## Trois fonctions disponibles

### ● Certificats de confiance

- Permet d'afficher la liste des autorités de certification enregistrées(celle initialement incluses et celles rajoutées)

### ● Identifiants de l'utilisateur (depuis Android 7.0)

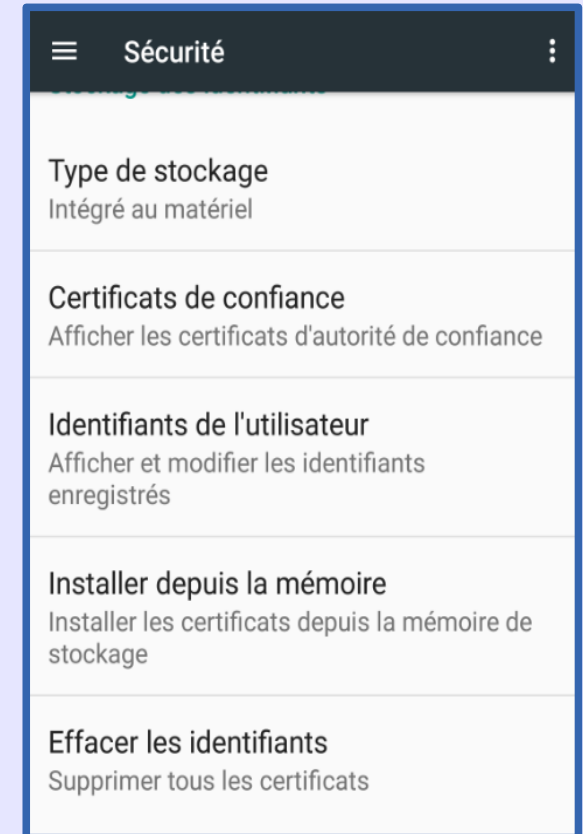
- Liste des certificats de l'utilisateur

### ● Installer depuis la mémoire

- Pour installer des certificats dans le magasin (y compris ceux des AC)
- Les certificats doivent être contenus dans des fichiers P12 ou CRT à la racine de la SDCARD.
- Après l'installation le fichier est détruit

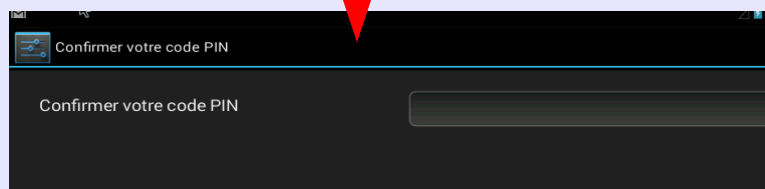
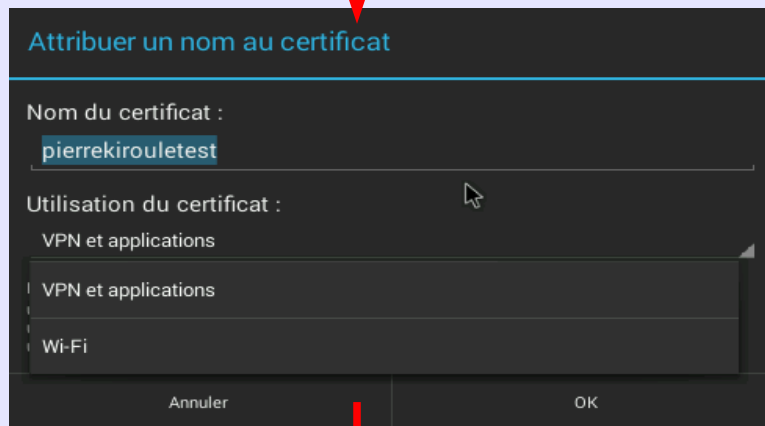
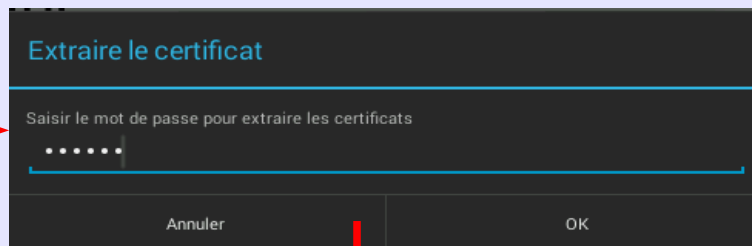
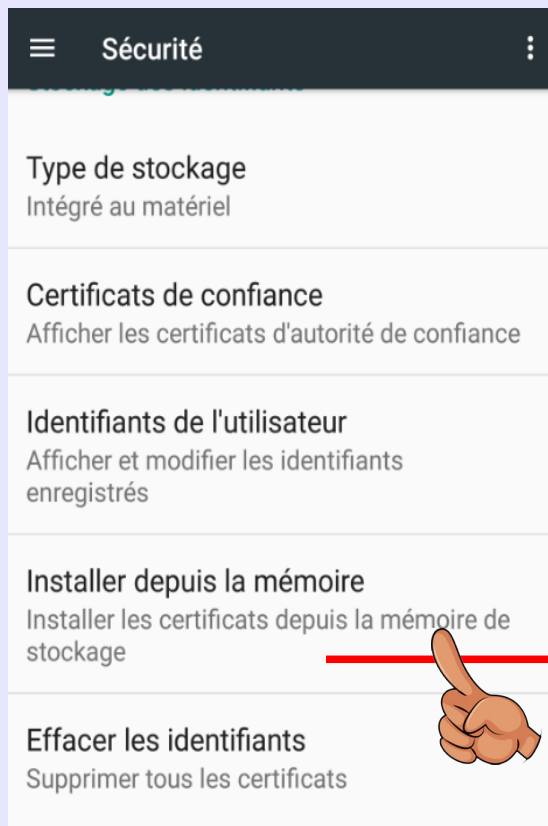
### ● Supprimer tous les certificats

- Supprime tous les certificats créés par l'utilisateur (on ne peut pas les supprimer individuellement)



# Les certificats sous Android : importation

- Paramètres/sécurité
- Installer depuis la mémoire

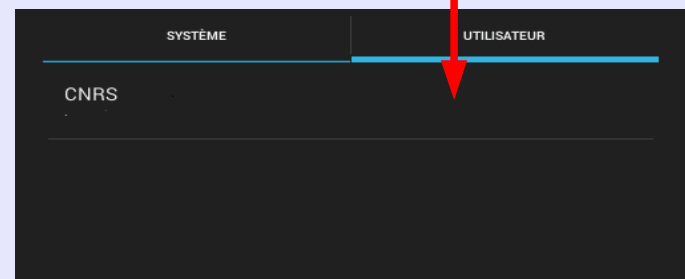
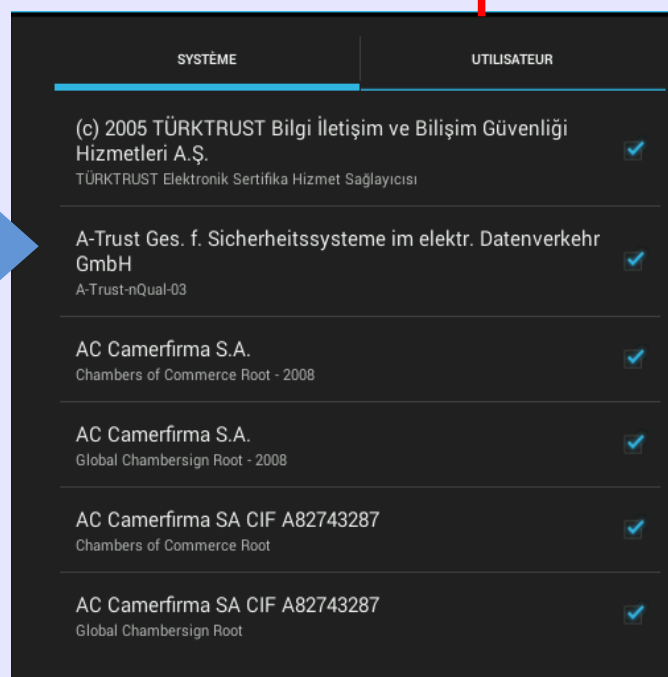
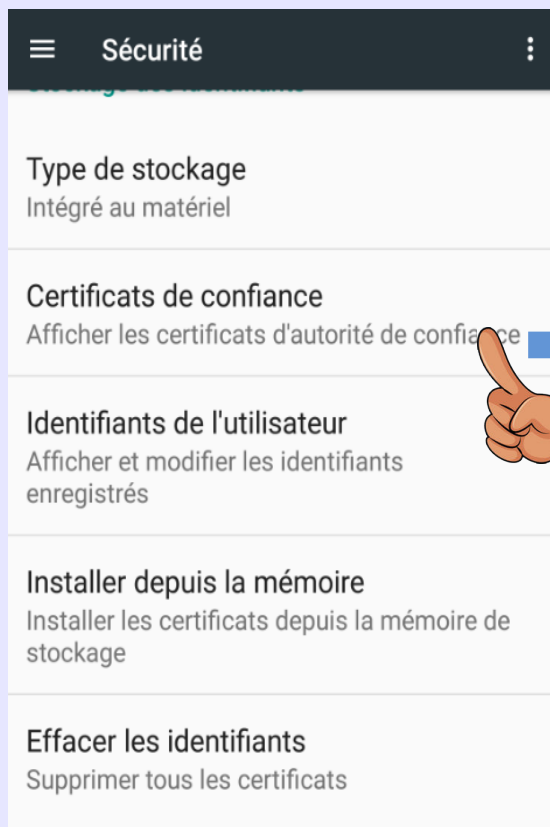


- Le mot de passe du fichier p12 est demandé
- Il ne sera plus utilisé par la suite
- Le nom du fichier est pris comme nom du certificat (alias). Il ne s'agit pas du CN mais d'un nom qui permettra ensuite de repérer le certificat.
- Ce nom peut être changé par ce que l'on veut.
- Il faut choisir le type de certificat (versions  $\geq 4.3$ )
- Le mot de passe demandé ici est le mot de passe de verrouillage du mobile pour permettre d'écrire dans le magasin.
- Le certificat est enregistré dans le magasin.

# Les certificats sous Android : importation

## Certificats des autorités de certification

- Le fichier P12 contenant un certificat peut aussi contenir le certificat de l'autorité de certification. Dans ce cas elle est importée en même temps.
- Pour importer spécifiquement une autorité, la procédure est identique au certificat utilisateur mais le fichier doit être du type **.crt**.



# Les certificats sous Android : pour le wifi (EAP/TLS)

## Pourquoi utiliser le protocole EAP/TLS ?

- Authentifier les mobiles avec leur certificat (et croiser avec l'adresse MAC : La connexion ne sera autorisée que si le certificat est présenté depuis un matériel qui possède l'adresse MAC enregistrée)
- Positionner les mobiles dans leur sous-réseau (pas forcément le même pour tous)
- Les mobiles authentifient aussi le certificat du serveur

L'authentification est réalisée par un serveur Radius avec une entrée du style :

CN du certificat

Adresse MAC du mobile

*pierre.dupont.mob1* Auth-Type := EAP , Calling-Station-Id == "*01:02:03:04:05:06*"  
Tunnel-type = VLAN,  
Tunnel-Medium-Type = 6,  
Tunnel-Private-Group-ID = *10* ← VLAN

# Les certificats sous Android : pour le wifi (EAP/TLS)

**Ajouter un réseau**

SSID du réseau: mywifi

Sécurité: 802.1x EAP

Méthode EAP: TLS

Authentification Phase 2: Aucun(e)

Certificat CA: mon.certif

Certificat utilisateur: mon.certif

Identité: Pierre.dupont.mob1

Enreg. Annuler

Le nom du SSID

Type de sécurité

Méthode

Certificat CA

Choix (dans la liste déroulante) du certificat à utiliser.

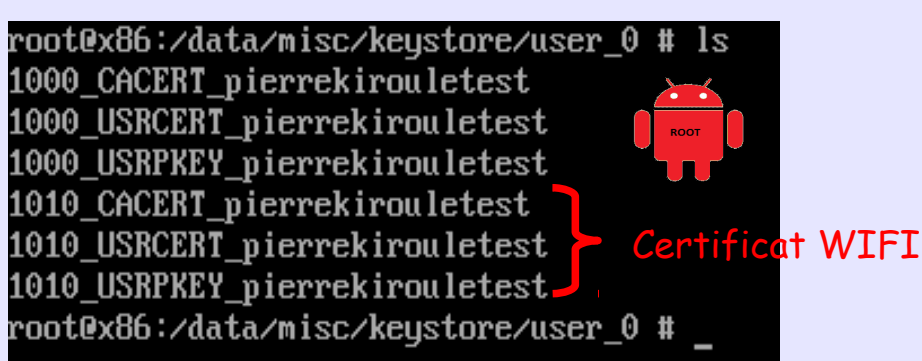
Le CN du certificat (c'est sur lui que se fera l'authentification)

# Les certificats sous Android : le keychain

- Sur un mobile non-rooté le keychain n'est accessible qu'au travers des API du service **keystore**.
- Les certificats sont stockés dans **/data/misc/keystore/user\_n** (n=uid de l'utilisateur)
- Accessible directement sur un mobile rooté ( => intérêt des MV pour tester)



```
root@x86:/data/misc/keystore/user_0 # ls
1000_CACERT_pierrekirouletest
1000_USRCERT_pierrekirouletest
1000_USRPKEY_pierrekirouletest
root@x86:/data/misc/keystore/user_0 #
```




```
root@x86:/data/misc/keystore/user_0 # ls
1000_CACERT_pierrekirouletest
1000_USRCERT_pierrekirouletest
1000_USRPKEY_pierrekirouletest
1010_CACERT_pierrekirouletest
1010_USRCERT_pierrekirouletest
1010_USRPKEY_pierrekirouletest
root@x86:/data/misc/keystore/user_0 #
```

- Ici un certificat type « VPN et applications » a été importé sous l'utilisateur « owner » : user\_0
- 1000 est l'UID du créateur (system)
- Le keystore peut être manipulé avec la commande (root) **keystore\_cli**
- Les autorités de certification ajoutées sont dans **/data/misc/keychain/cacerts-added**

# Les certificats sous Android : le keychain

- Lorsque la fonction multi-utilisateur est utilisée, chaque utilisateur possède ses propres certificats
- Le répertoire keystore contient en fait un sous-répertoire par utilisateur.

```
root@x86:/ # cd /data/misc/keystore/  
root@x86:/data/misc/keystore # ls  
user_0  
user_10  
user_11  
root@x86:/data/misc/keystore #
```



```
root@x86:/data/misc/keystore/user_11 # ls  
1101000_CACERT_jeanpeuplustest  
1101000_USRCERT_jeanpeuplustest  
1101000_USRPKEY_jeanpeuplustest  
root@x86:/data/misc/keystore/user_11 #
```



- Les utilisateurs autre que « Owner » ne peuvent pas importer de certificats type Wifi



# Les certificats sous Android : le keychain

---

- Le Keychain d'Android ne stocke **que des certificats**.
- Si le mobile ne supporte pas ARM TrustedZone, la clé privée des certificats est chiffrée avec une master key générée aléatoirement et chiffrée avec le passcode de l'utilisateur (**/data/misc/keystore/user\_n/.masterkeyfile**)  
Voir : <http://nelenkov.blogspot.fr/2011/11/ics-credential-storage-implementation.html>
- Si le mobile supporte ARM TrustedZone la clé privée des certificats est chiffrée par une clé contenue dans le processeur (TEE)

---

## Certificats sous iOS

---



# Les Certificats sous iOS

---

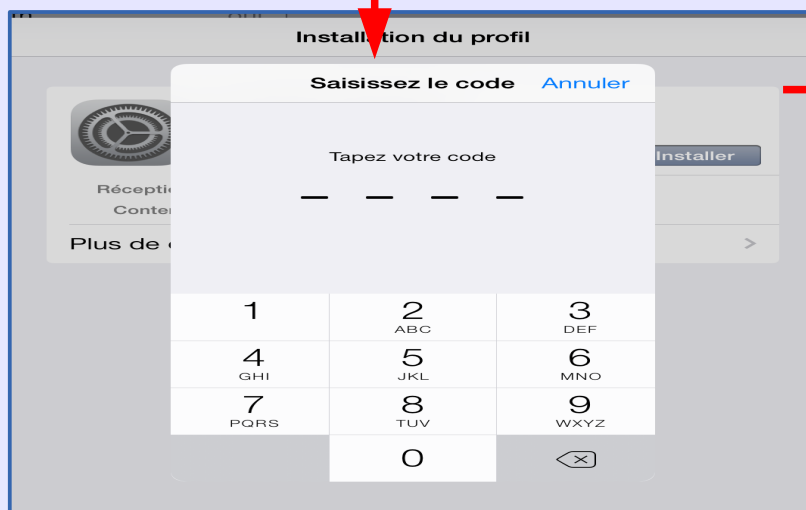
- Contrairement à Android il n'est pas possible de copier un fichier P12 dans le système de fichiers d'iOS (même avec iTunes)
- Il faut donc obligatoirement transférer le certificat avec un navigateur connecté à un serveur via le Wifi.
- C'est embêtant quand il faut justement le certificat pour se connecter au Wifi

# Les Certificats sous iOS : importation

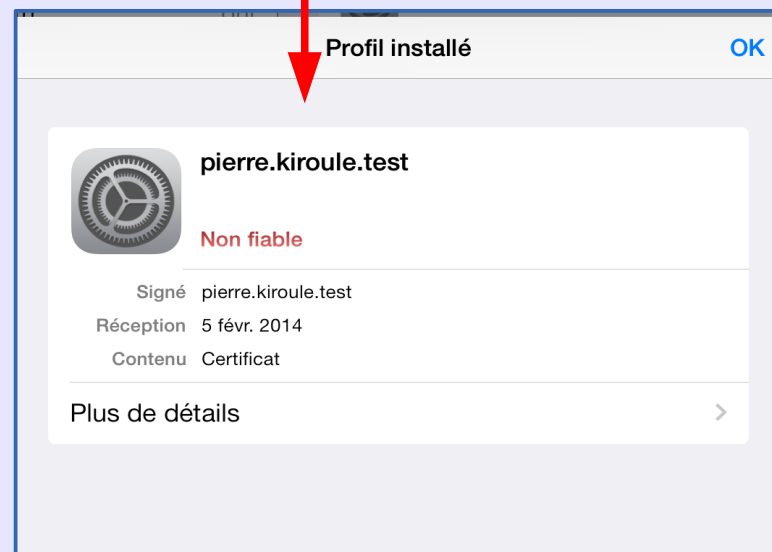
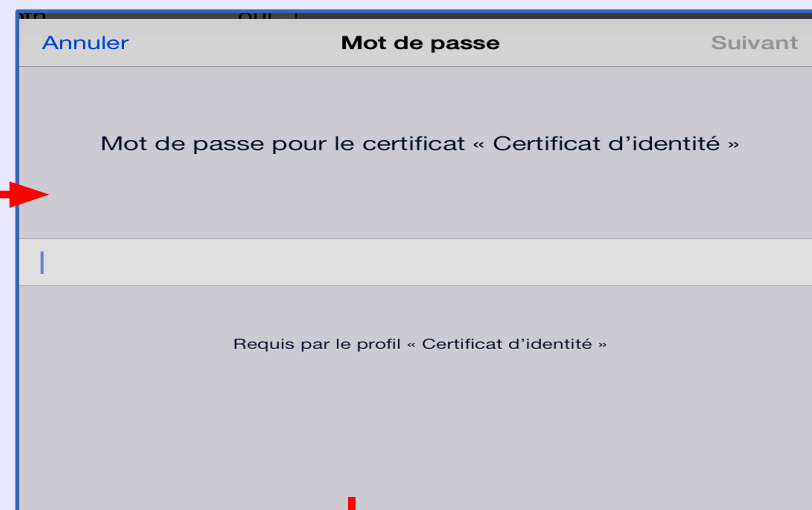
- Lorsque safari télécharge le fichier P12, l'installation du certificat est automatiquement déclenchée



- Le pincode de verrouillage est demandé

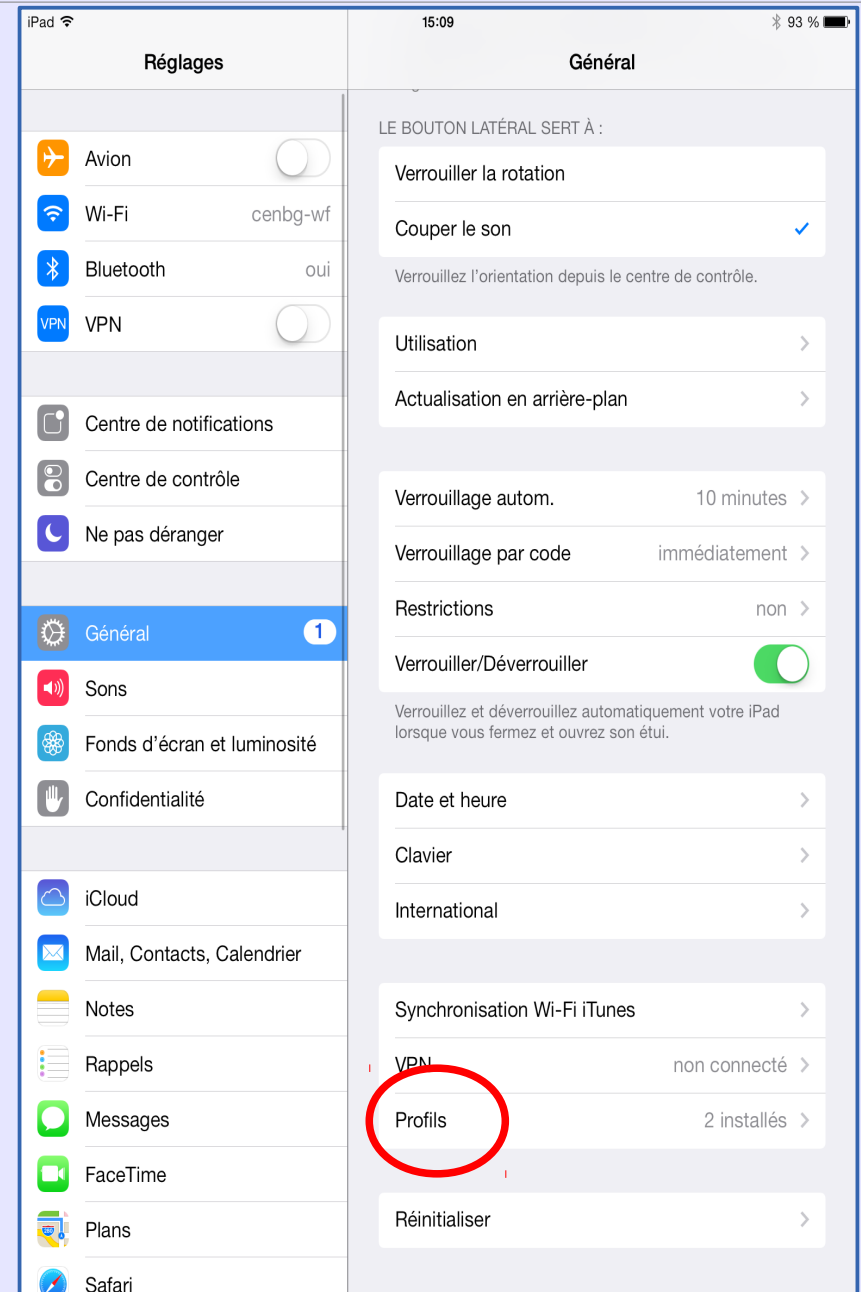


- Le mot de passe d'importation du fichier P12 est demandé

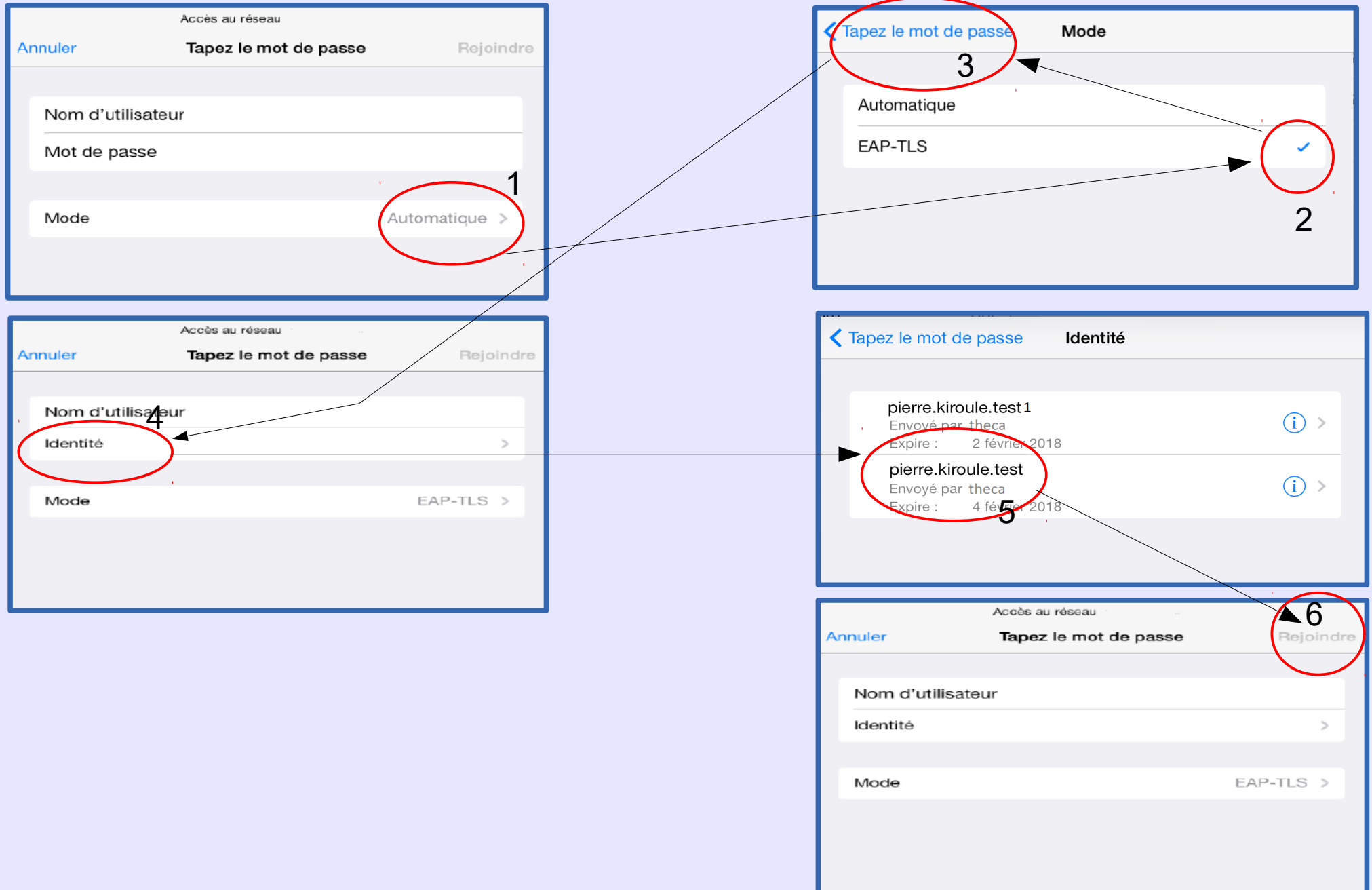


# Les Certificats sous iOS : importation

- Les certificats peuvent être visualisés dans les réglages.
- Ils peuvent être listés et détruits individuellement.



# Les certificats sous iOS : pour le wifi (EAP/TLS)



---

## Partage de connexion : mode modem

---

# Partage de connexion : mode modem

---

## A quoi ça sert ?

Permet de connecter un ordinateur sur Internet via un smartphone équipé de la 3G (ou mieux)

## Comment ?

- En connectant le smartphone sur l'ordinateur en **USB** (il suffit de connecter le câble USB)

Ou bien

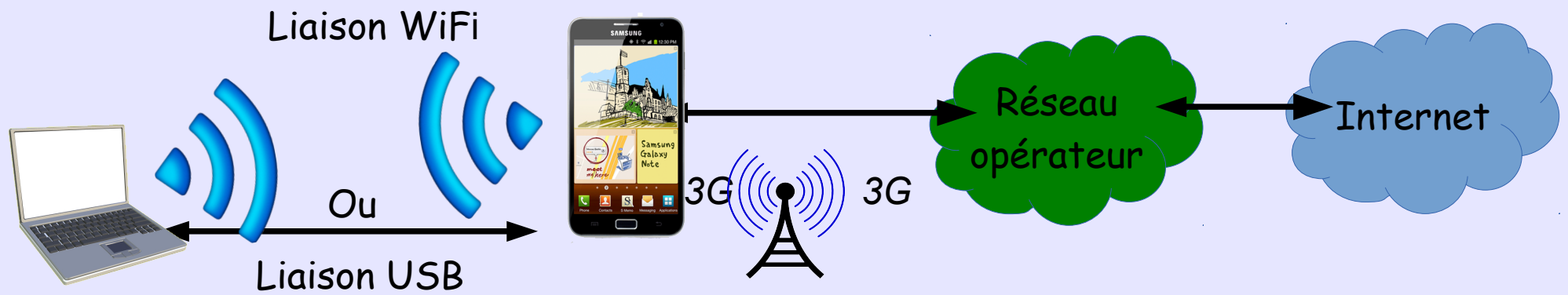
- En se servant du smartphone comme une **borne WiFi** sur laquelle l'ordinateur se connecte

## Condition ?

- Le smartphone dispose d'une connexion DATA
- L'opérateur autorise le mode modem



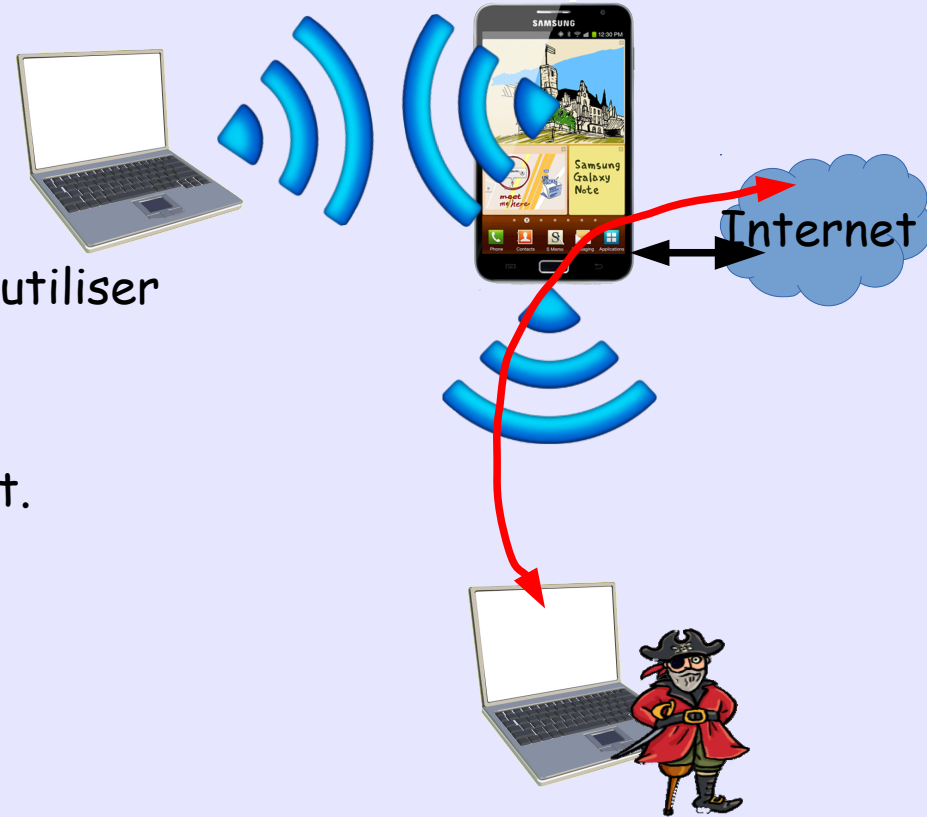
# Partage de connexion : mode modem



# Partage de connexion : mode modem

## Une tierce personne peut-elle utiliser la connexion Internet du smartphone ?

- Le smartphone diffuse comme une borne Wifi
- Tous ceux qui sont à proximité captent le SSID
- Si la sécurité est « ouvert », tout le monde peut utiliser le réseau WiFi créé par le smartphone.
- Idem si le mot de passe est connu ou trop évident.



# Partage de connexion : mode modem

## Y a-t-il un risque d'interconnexion du réseau interne avec l'extérieur ?



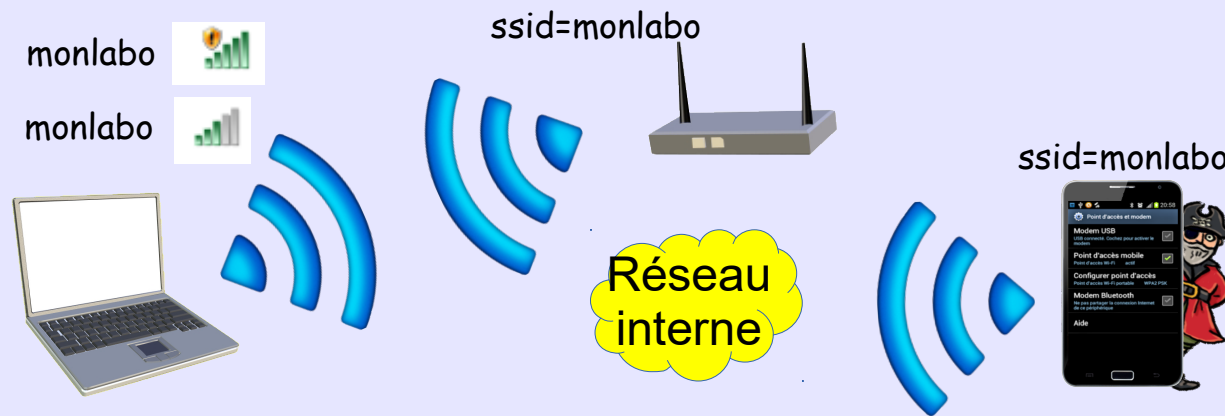
- Cela suppose que le pirate trouve le mobile à travers Internet et le réseau de l'opérateur et qu'il soit en mode modem à ce moment-là.
- Cela suppose que le smartphone route les paquets vers l'ordinateur connecté
- Cela suppose que l'ordinateur route les paquets vers sa liaison Ethernet libre vers une cible potentiel.
- Compte tenu que le réseau ordinateur-smartphone est éphémère cela semble très compliqué, en tout cas sans une « aide » sur l'ordinateur et ou smartphone (genre virus par exemple).

# Partage de connexion : mode modem

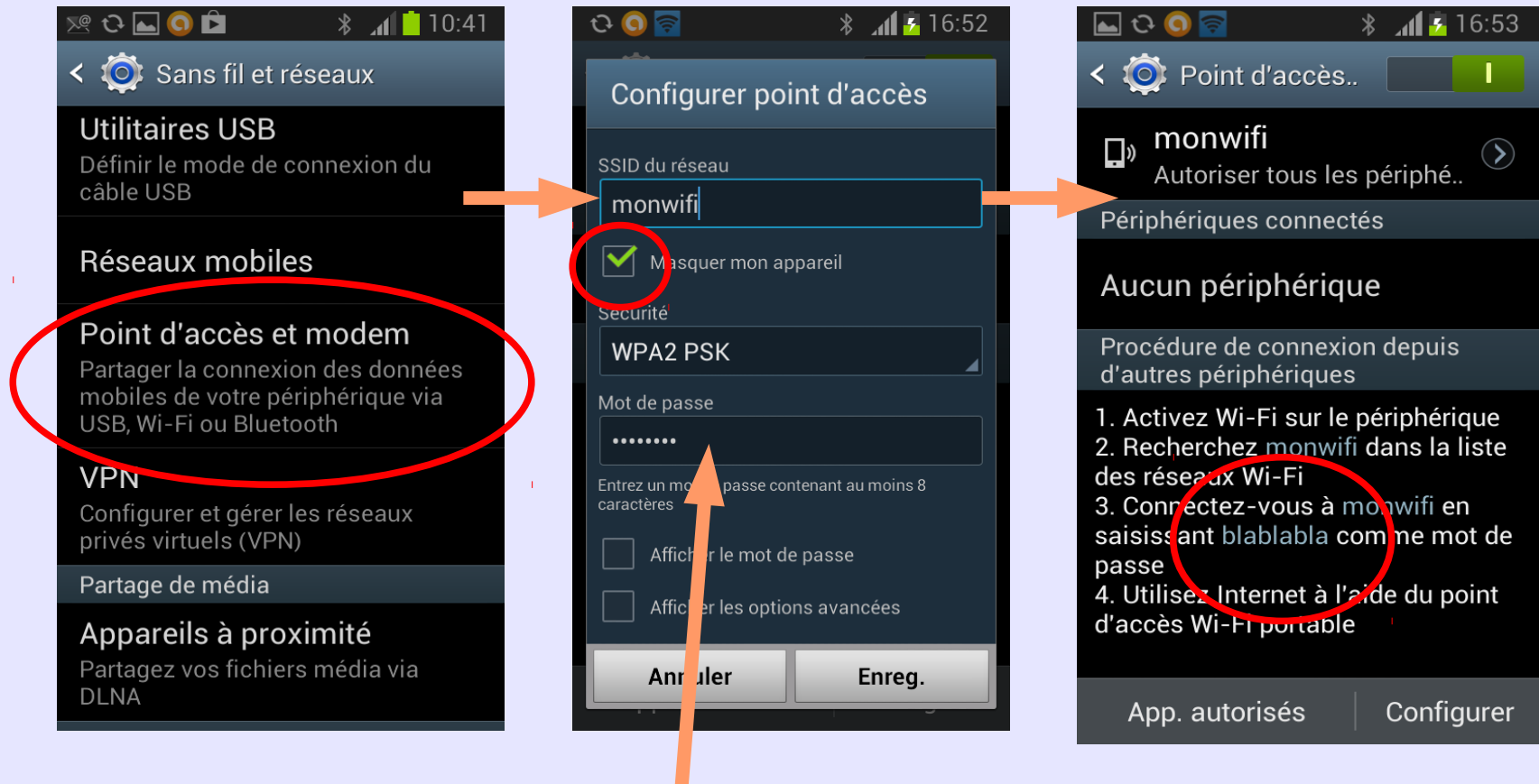
## Y a-t-il un risque de piratage d'un SSID du réseau local?

C'est probablement le plus gros risque du mode modem !!

- N'importe qui peut configurer un point d'accès avec un SSID, ouvert, de même nom qu'un SSID existant. C'est pas nouveau, mais beaucoup plus facile et accessible sans grande compétence.
- Un utilisateur, sur n'importe quel ordinateur WiFi, verra deux SSID de même nom. S'il sélectionne le mauvais il se connectera au réseau via le mobile pirate qui pourra agir comme « mobile-in-the-middle »

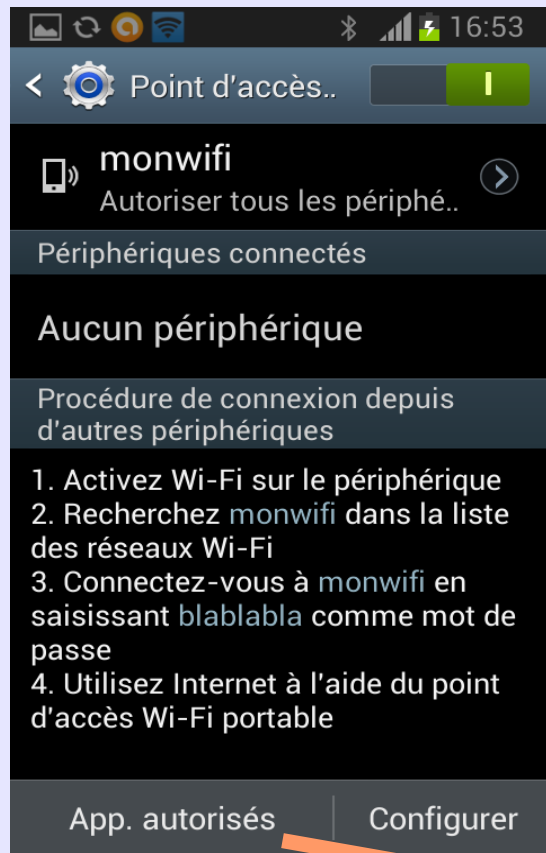


# Partage de connexion : configuration

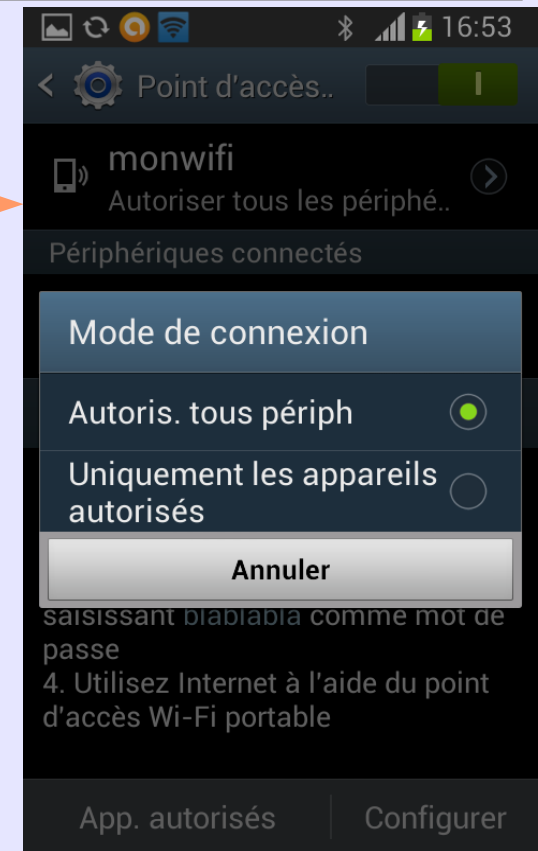


- **SSID** : Ne pas utiliser son propre nom ou un SSID existant
- **Sécurité** : Ne pas utiliser une sécurité « ouverte »
- **Mot de passe** : ne pas utiliser le mot de passe d'un compte (il est stocké en clair)

# Partage de connexion : configuration



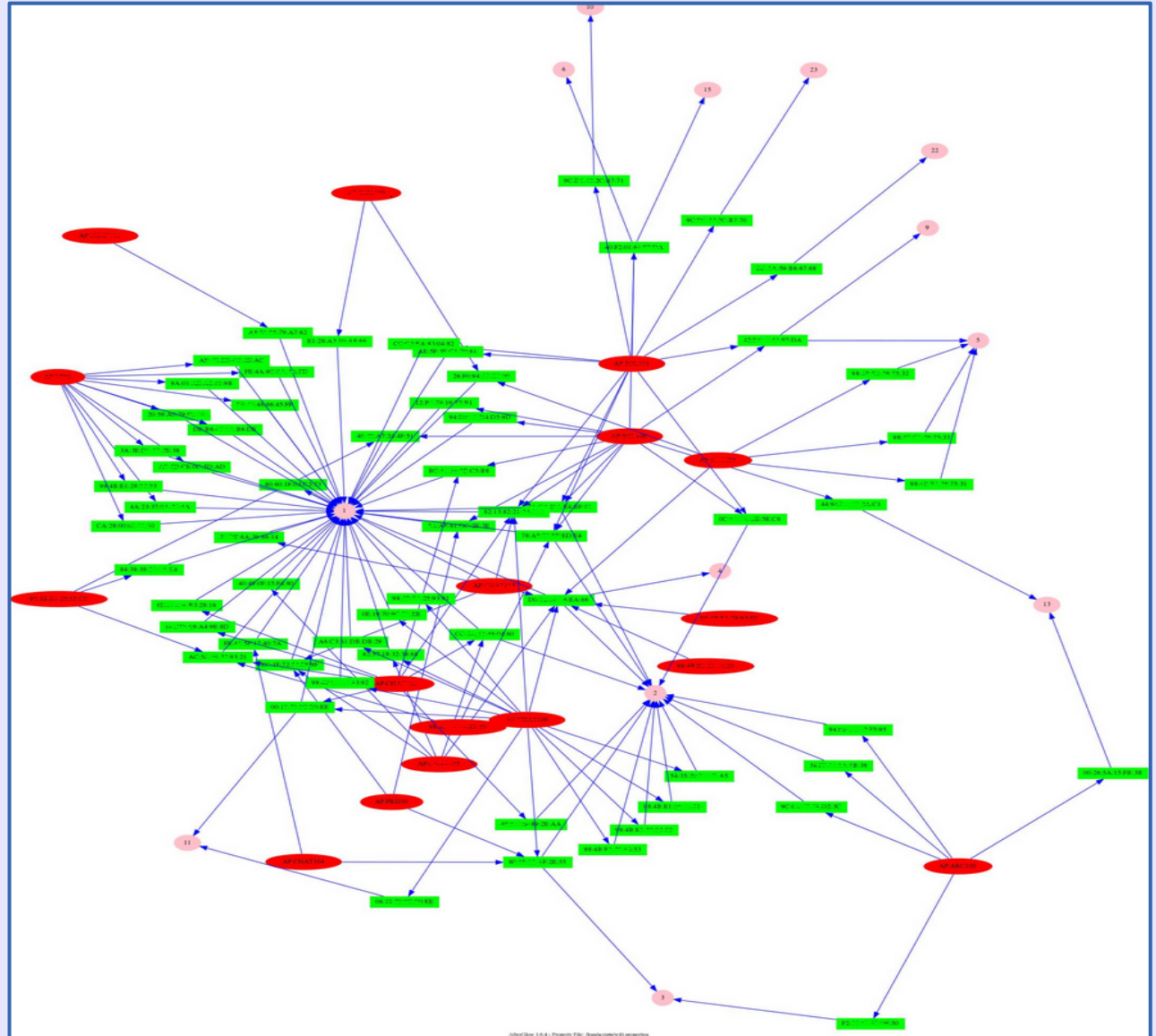
De préférence autoriser uniquement ses propres matériels



# Partage de connexion : et « ils » s'en servent

Repérage sur un mois des mobiles en mode partage de connexion c'est-à-dire diffusant un SSID sur un réseau de laboratoire.

- Bornes officielles du réseau
- Mobiles diffusant un SSID (rogue AP)
- Nombre de fois qu'un mobile a été repéré



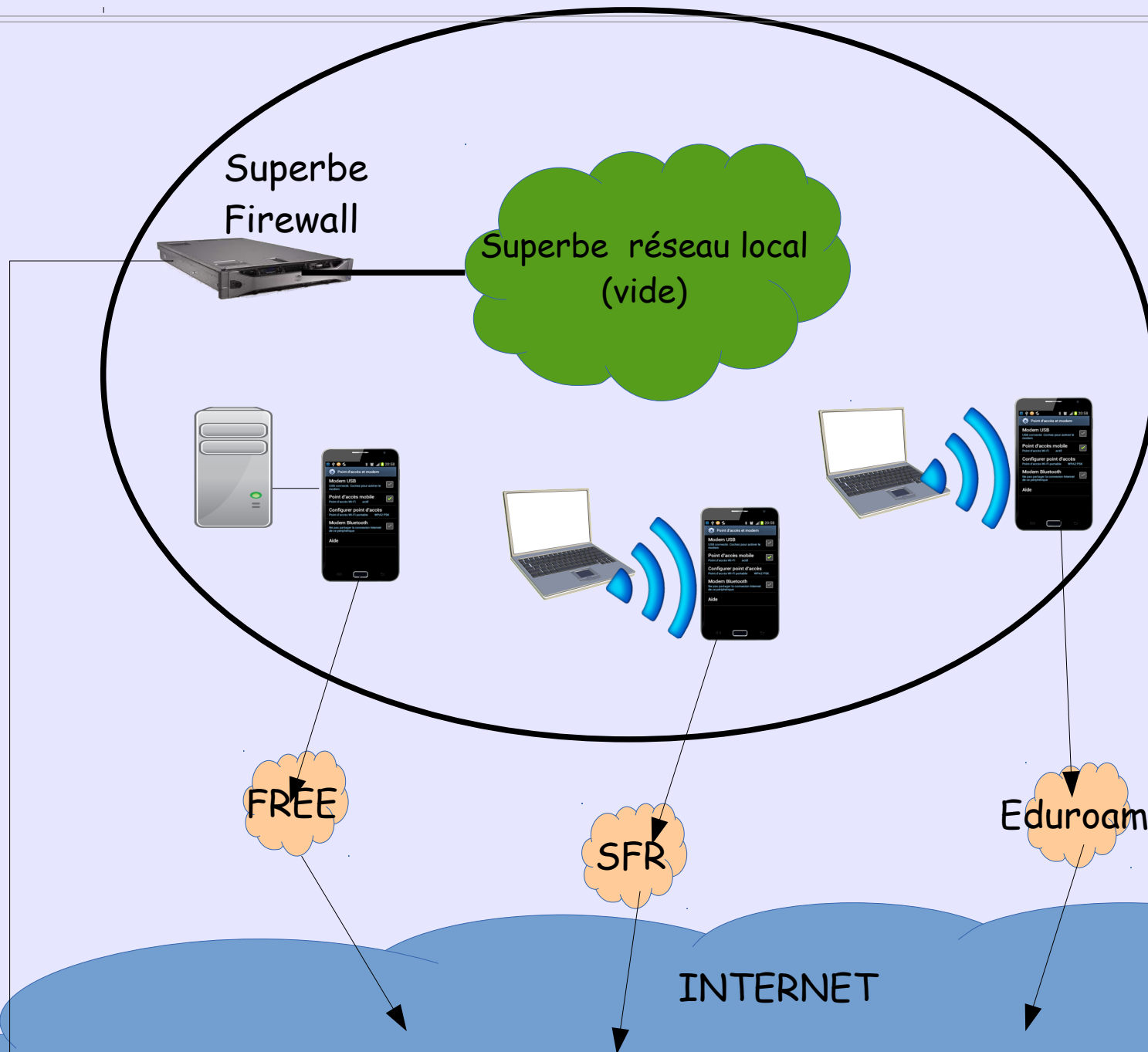
# Accueil Wifi

---

- Tous les utilisateurs ont configuré Eduroam dans leur smartphone... et pas forcément correctement (voir plus loin).
- Eduroam favorise l'utilisation de matériels personnels, non connus, non enregistrés....
- L'utilisation du partage permet de connecter n'importe quelle machine (y compris fixe via USB) directement sur le réseau des opérateurs => l'utilisateur n'a plus vraiment besoin du réseau du labo !
- Les promesses de la 5G (> 1Gbs) vont accentuer ce phénomène et ringardiser les Wifi locaux.

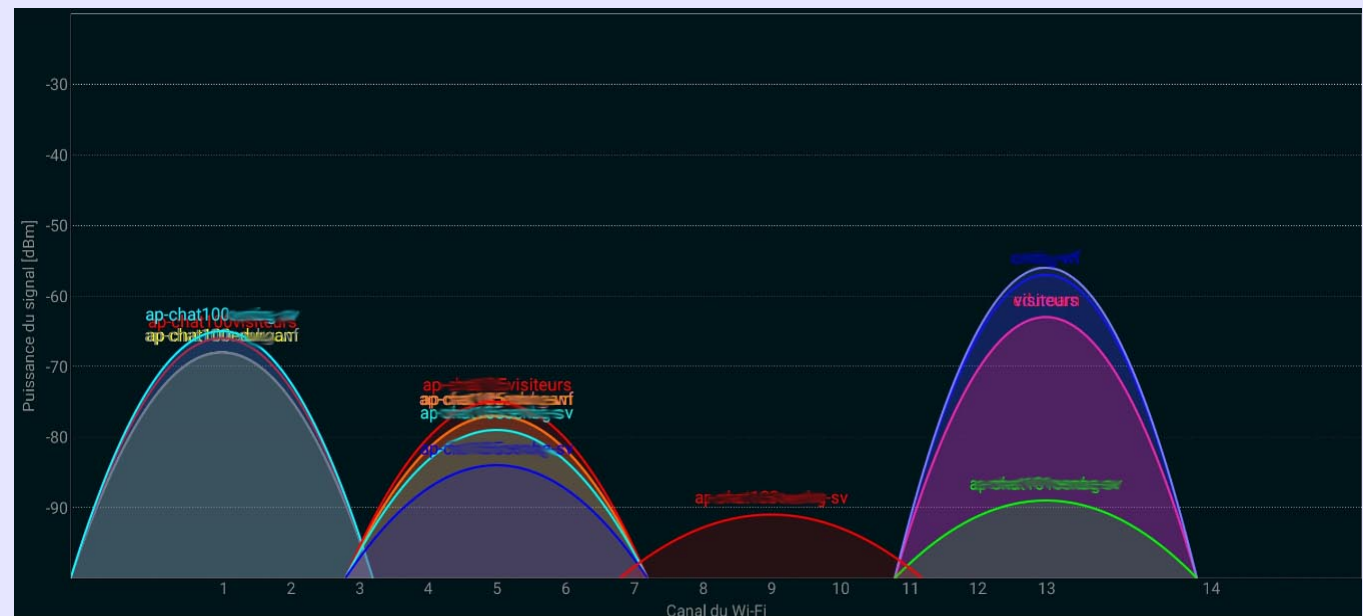


# Le futur réseau local d'un labo?



# Wifi

La multiplication des smartphones en mode partagé peut saturer la bande des 2,4Ghz et provoquer des superpositions de canaux avec les bornes officielles et des perturbations.

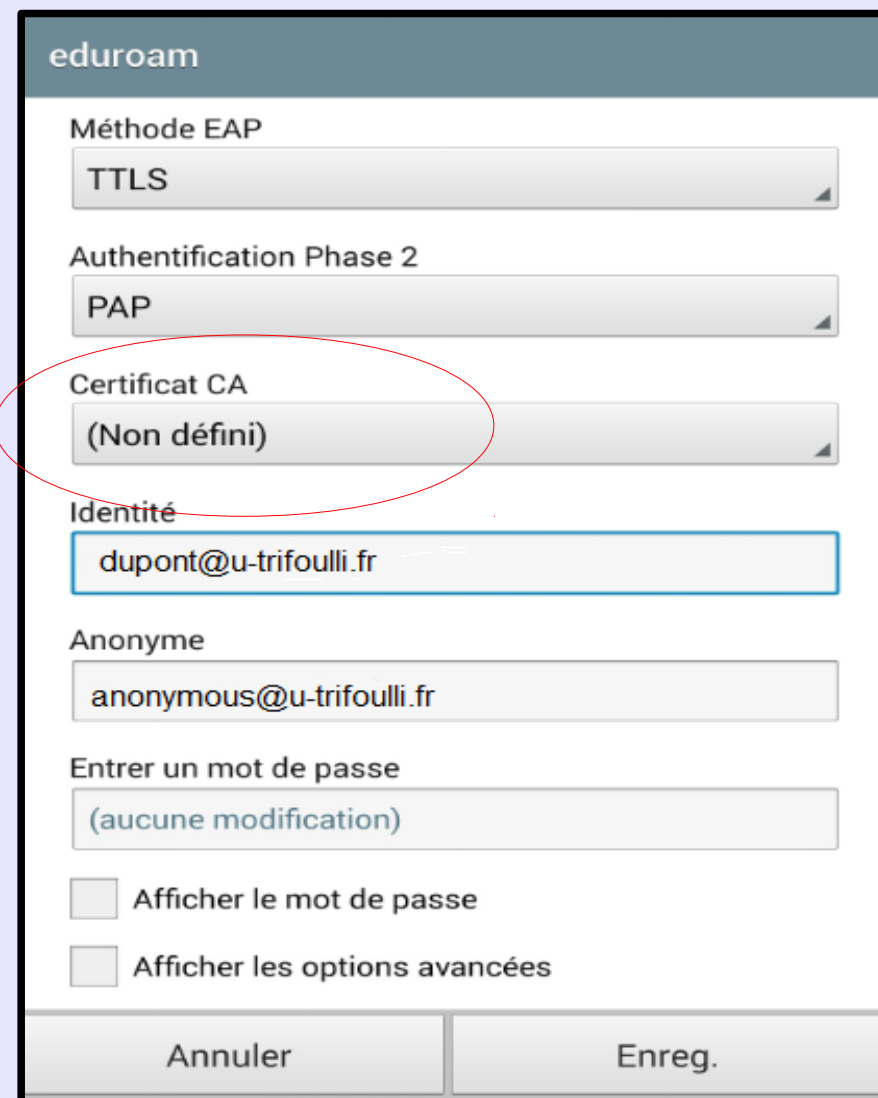


Visualisation de l'occupation des canaux avec l'application Wifi Analyzer

- Certains iPhone perturbent les communications Wifi au point de provoquer la déconnexion d'autres matériels
- Le Bluetooth intégré des Mac (souris par exemple) provoque des coupures du Wifi de la machine et éventuellement d'autres Mac proches.
- Certains constructeurs (Sony par exemple) équipe les alimentations de leur PC portable de borne Wifi qui émettent sans arrêt. (L'alimentation possède également un interface filaire et peut ainsi permettre au PC de se connecter en filaire via la borne Wifi).

# Eduroam : comment MAL le configurer...

- Sous Android il est possible de configurer Euduroam « à la main », sans remplir le champ « Certificat CA » comme ici :
- Ce que font beaucoup d'utilisateurs.
- Si un faux SSID Eduroam est émis par une borne pirate adossée à son propre serveur Radius, le mot de passe peut être facilement tracé.
- Comme le Wifi se connecte automatiquement au SSID qu'il connaît, l'utilisateur ne verra rien et pourtant son mot de passe aura été volé.



eduroam

Méthode EAP  
TTLS

Authentification Phase 2  
PAP

Certificat CA  
(Non défini)

Identité  
dupont@u-trifoulli.fr

Anonyme  
anonymous@u-trifoulli.fr

Entrer un mot de passe  
(aucune modification)

☐ Afficher le mot de passe

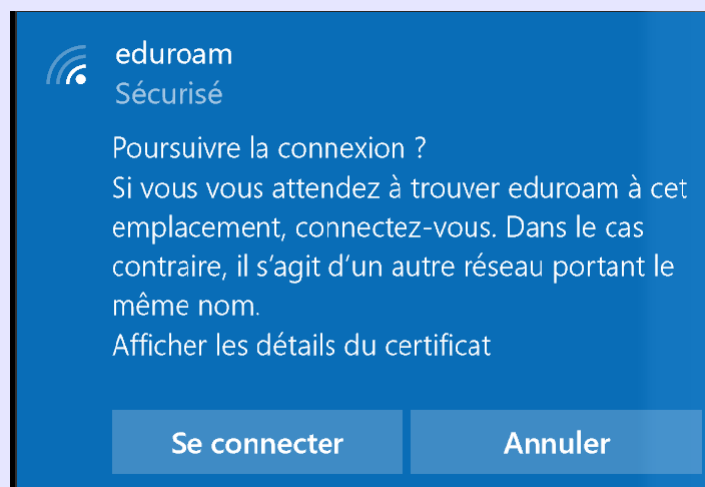
☐ Afficher les options avancées

Annuler Enreg.

# Eduroam : comment mal le configurer...

## Même problème avec Windows 10 (et d'autres)

- Lors de la première connexion, le système enregistre bien le certificat du serveur Radius. L'utilisateur verra la fenêtre suivante :



- Mais si un jour l'utilisateur est confronté à un faux SSID Eduroam, donc un autre serveur Radius, il verra la même fenêtre et cliquera inmanquablement sur « Se connecter » et son mot de passe pourra être tracé en clair sur le serveur Radius pirate.

# Eduroam : comment mal le configurer...

## Sur le serveur radius Pirate

- Fichier **users** :

**DEFAULT Auth-Type:= EAP**

**DEFAULT Auth-Type:= ACCEPT**

- Lancement en mode debug : `radiusd -X`

```
.  
.  
(9) eap_ttls: Session established. Proceeding to decode tunneled  
attributes  
(9) eap_ttls: Got tunneled request  
(9) eap_ttls: User-Name = "dupont@mon-univ.fr"  
(9) eap_ttls: User-Password = "motdepasseenclair"  
(9) eap_ttls: Sending tunneled request  
(9) Virtual server inner-tunnel received request  
(9) User-Name = "dupont@mon-univ.fr"  
(9) User-Password = "motdepasseenclair"  
.  
.
```

# Eduroam : comment BIEN le configurer...

## Sous Android

- Il faut installer l'application Eduroam-CAT
- Sélectionner son organisme de rattachement
- Entrer son identifiant et mot de passe

eduroam

TTLS

Authentification étape 2

PAP

Certificat CA

eduroam\_WPA\_EAP\_TTLS\_...

Identité

dupont@mon-univ.fr

Anonyme

anonymous@mon-univ.fr

Mot de passe

(aucune modification)

☐ Afficher le mot de passe

Options avancées

ANNULER ENREGISTRER

# Applications à risques

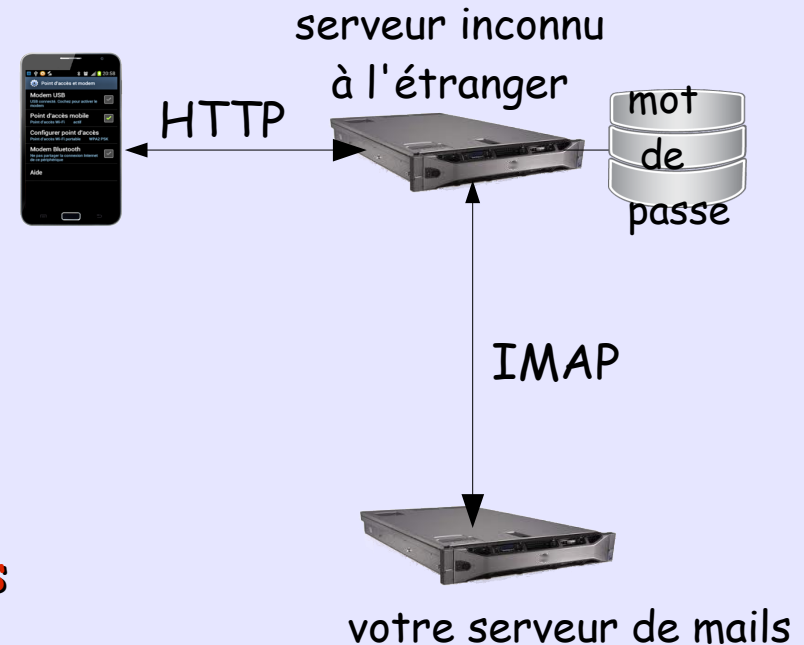
Diverses applications utilisent des serveurs tiers par lesquels transitent les données et où sont stockés les mots de passe.

- **Applications de mail** qui stockent le mot de passe sur un serveur qui lui même relève le courrier sur le serveur de mail.

(mymail, cloumagic, bluemail...)

<http://labs.alinto.com/1570/les-pieges-dune-appli-android-mymail/>

**Le mot de passe est compromis**





# Impressions via un serveur d'impressions local

## Let's print Droid - Configuration

Printer Name:  
lp0

Protocol:

- SMB - Windows Shared Printer
- GCP - Google Cloud Print
- IPP - IPP/CUPS print server
- IPPS - IPP/CUPS SSL server
- HTTP - Web Server POST
- HTTPS - encrypted POST
- FILE - CIFS/SMB File drop
- FTP - File Transfer
- FTPS - File Transfer SSL
- SHARE - Share PDL with another app
- VIEW - View PDL with another app

Printer Name:  
lp0

Protocol:  
IPP - IPP/CUPS print server

IP Address/Computer Name  
borimp1.cenbg.in2p3.fr

Port Number  
631

Queue/Share/Dir Name  
/printers/lp0

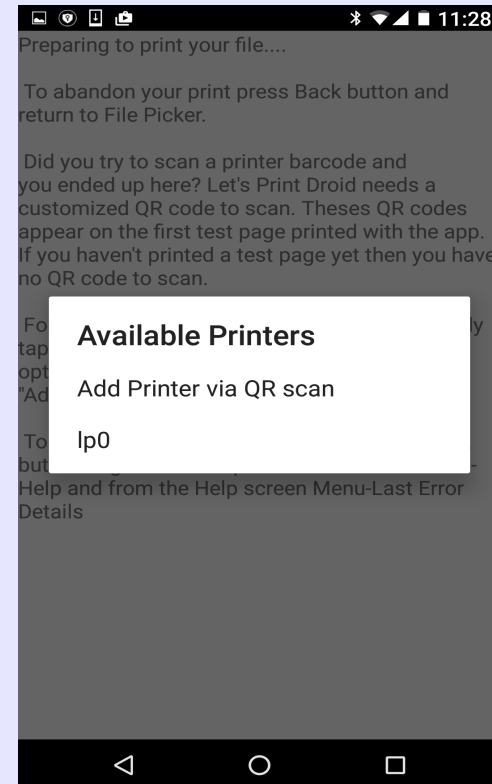
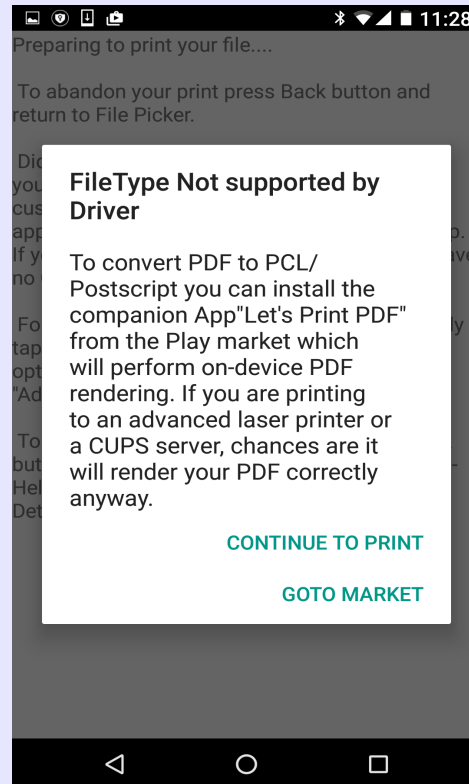
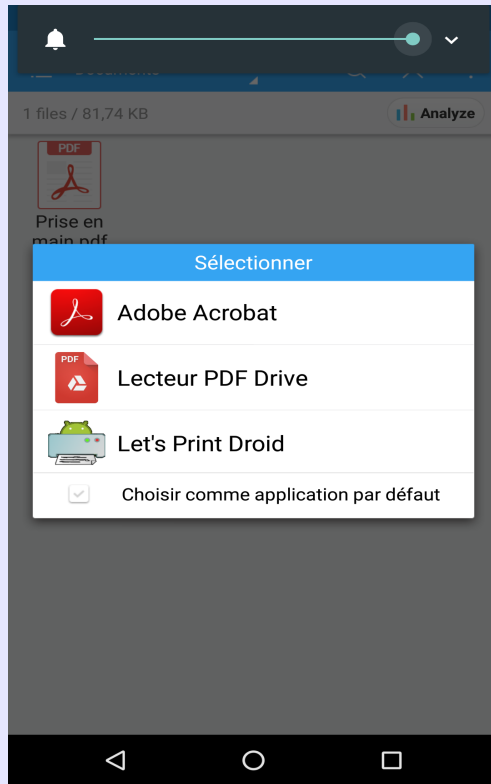
Page Description Language:  
PS - Postscript

☐ Print a test page/QR code scan sheet

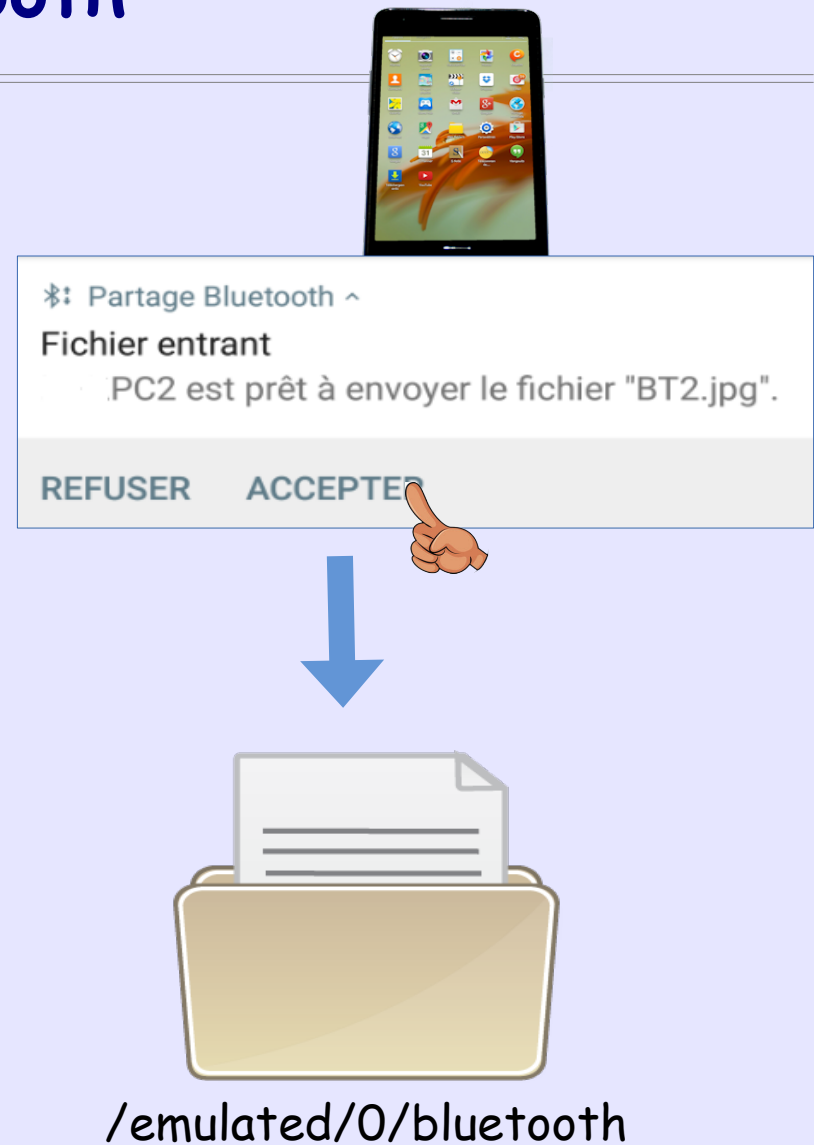
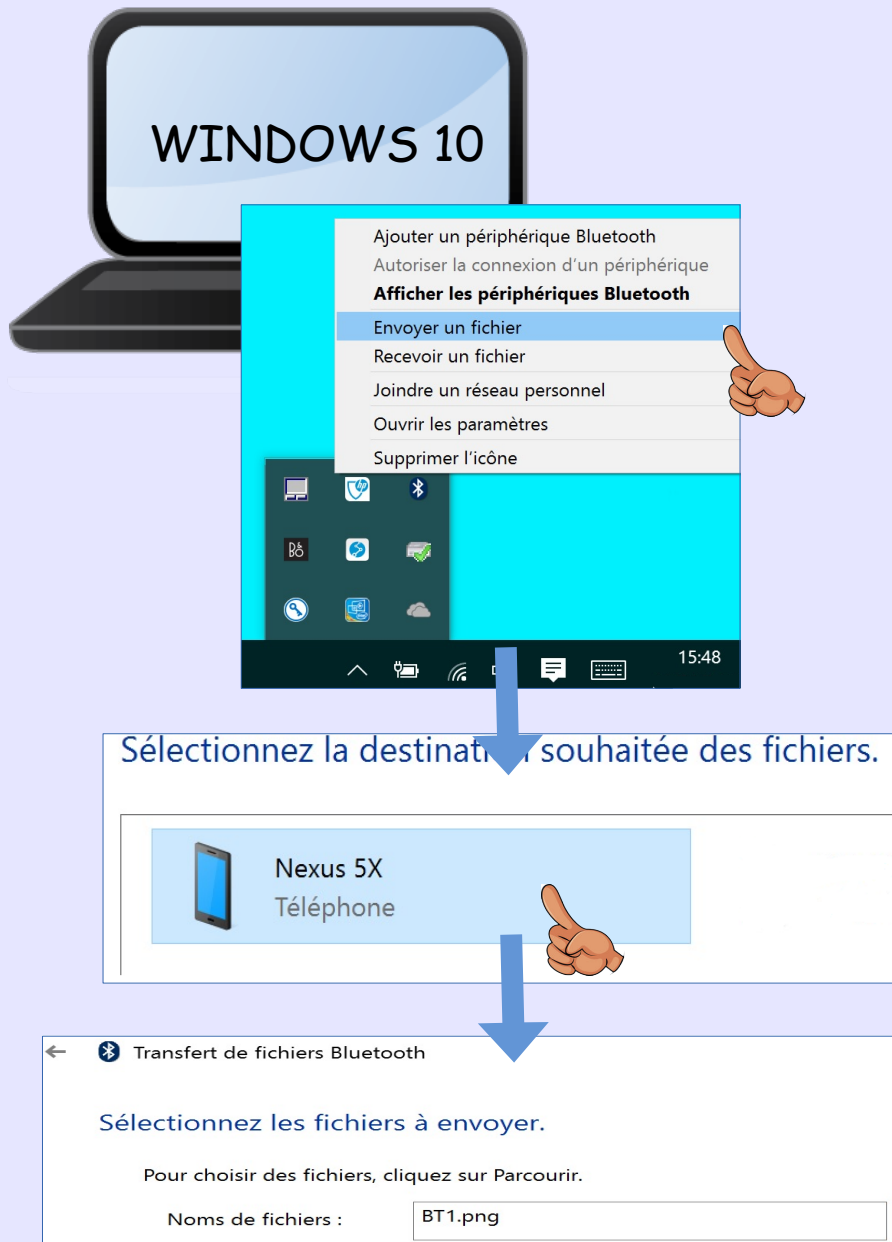
SAVE CHANGES

# Impressions via un serveur d'impressions local

## Exemple d'impression

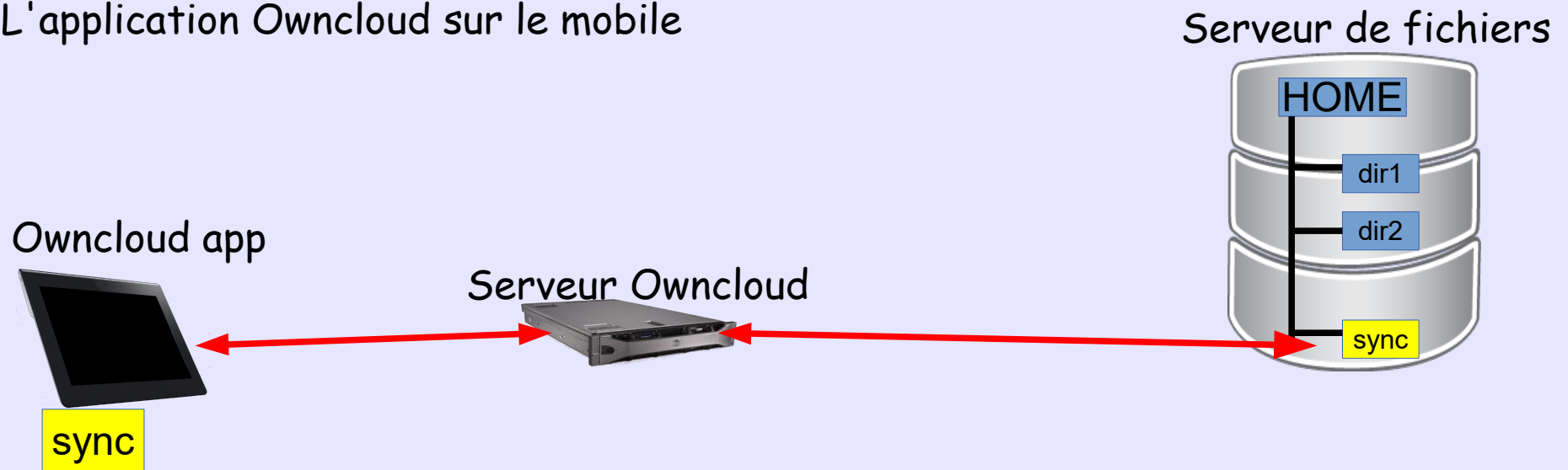


# Transfert de fichiers via Bluetooth



# Synchronisation de fichiers avec Owncloud

- Permet de synchroniser un répertoire distant avec le mobile
- Il faut un serveur Owncloud qui a accès à l'espace utilisateur (home)
- L'application Owncloud sur le mobile



# Synchronisation de fichiers avec Owncloud

- Sur le serveur de fichiers  
l'utilisateur définit un répertoire de synchronisation (exemple SYNC)
- Sur le serveur **Owncloud**  
l'administrateur autorise le stockage externe pour les utilisateurs
- Sur le serveur **Owncloud**  
l'utilisateur se connecte avec son compte et définit un stockage externe

## Stockage externe

Nom du dossier

Stockage externe

Configuration

Nom du dossier

Ajouter un support

☒ Activer le stockage externe pour les utilisateurs

Autoriser les utilisateurs à monter les stockage externes suivants

☐ Compatible avec Amazon S3

☐ Dropbox

☐ FTP

☐ Google Drive

☐ Object de Stockage OpenStack

☒ ownCloud

☐ SFTP

☒ SMB / CIFS

☐ SMB / CIFS utilise le nom d'utilisateur OC

☐ WebDAV

**Stockage externe**

Nom du dossier	Stockage externe	Configuration
<div><div></div><div>SYNC</div></div>	<div>SMB/CIFS</div>	<div>Filer.labo.fr</div> <div></div> <div></div> <div>Dupont/SYNC</div> <div>/</div>
<div>Nom du dossier</div>	<div>Ajouter un support</div>	

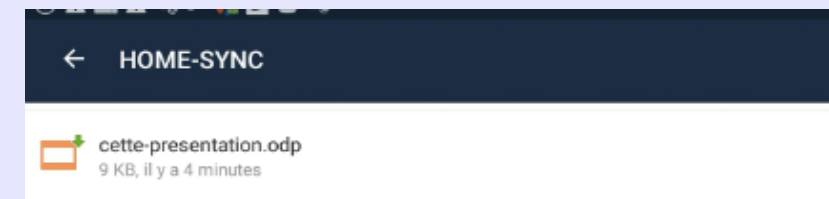
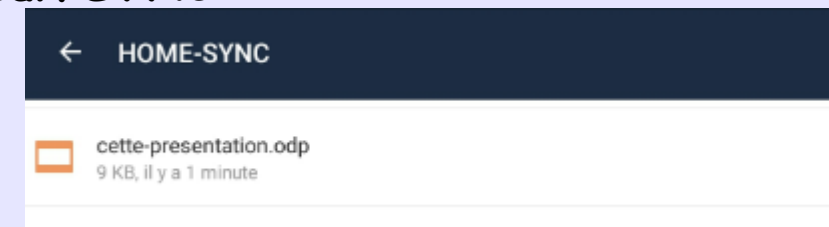
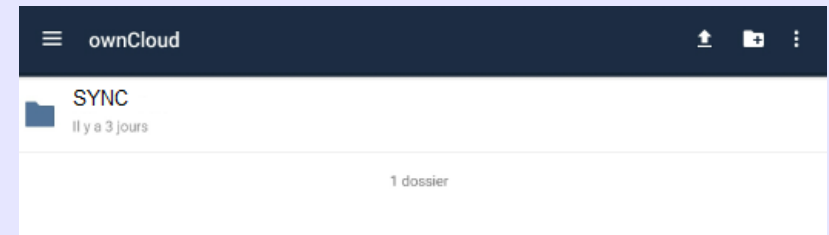
# Synchronisation de fichiers avec Owncloud

## Sur le mobile

- L'utilisateur installe l'appli et la lance
  - Il s'authentifie sur le serveur
  - Il voit le répertoire qu'il a défini sur le serveur
  - Seule la liste des fichiers est téléchargée
  - Pour synchroniser un fichier, il clique dessus
  - Le fichier apparaît alors une flèche verte
- 
- Les fichiers locaux se trouvent dans :  
**/Stockage de l'appareil/owncloud/dupont@serveur/SYNC**



The login screen for Owncloud. At the top is the Owncloud logo. Below it is a text input field for the server address, pre-filled with "Adresse du serveur https://...". Underneath are two more text input fields for "Nom d'utilisateur" and "Mot de passe". A blue "CONNECTER" button is positioned below the password field. At the bottom, there is a link that says "NOUVEAU DANS OWNCLOUD ?".



---

## Connexion VPN depuis un mobile

---

# Connexion VPN depuis un mobile (OpenVpn)

---

## Intérêts :

- La machine cliente est vue comme si elle était sur le réseau interne
- Possibilités de filtrage
- Possibilité d'authentification à double facteur
- L'application cliente OpenVpn est disponible sous Android et iOS



## Authentifications possibles

- Avec un simple mot de passe (pas recommandé)
- Avec un certificat et un mot de passe
- Avec un TOTP type Google Authenticator et un mot de passe.



# Connexion VPN depuis un mobile (OpenVpn)

---

## Que faut-il pour une authentification certificat+mot de passe ?

- Une IGC
- Un serveur Radius
- Un serveur OpenVpn
- L'application **Openvpn Connect** côté client → disponible sur Android et iOS.  
(ou Openvpn sous Windows/Linux, Tunnelblick sous MacOS)

**Le même dispositif est utilisé par tous les types de clients**

# Connexion VPN depuis un mobile (OpenVpn)

## Installation/configuration côté serveur OpenVpn

Dans cet exemple **radius.sh** et **password.sh** sont des scripts maison qui permettent de faire ce que l'on souhaite

**Radius.sh** est un script qui interroge le serveur Radius pour autoriser ou pas le client suivant :

- Mot de passe
- CN du certificat
- Adresse MAC du client (Openvpn >= 2.4 - à tester)

**password.sh** est un script qui peut permettre de faire des contrôles sur le mot de passe (et tout ce qu'on voudra)

```
local 192.168.0.10
port 1194
proto tcp
tmp-dir /tmp
dev tun
script-security 2
comp-lzo yes
keepalive 10 120
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
client-connect /path/.../radius.sh
auth-user-pass-verify /path/.../password.sh
dh /etc/openvpn/dh2048.pem
cipher AES-256-CBC
auth SHA512
server 10.8.0.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
persist-key
persist-tun
status /var/log/openvpn-status.log 20
log /var/log/openvpn.log
```

# Connexion VPN depuis un mobile (OpenVpn)

## Installation/configuration côté serveur OpenVpn et Radius

### Comment interroger un serveur Radius depuis Openvpn ?

- Soit en utilisant Pam\_radius → mais très limité
- Soit en dans radius.sh

```
echo "User-name=leCNducertif, User-Password=lepassword, Calling-Station-Id=00:01:02:03:04:05"  
| radclient -r 2 -x server-radius auth secret 2&> /path/.../reponse
```

On peut passer n'importe quel request-item au serveur Radius

### Et sur le serveur Radius

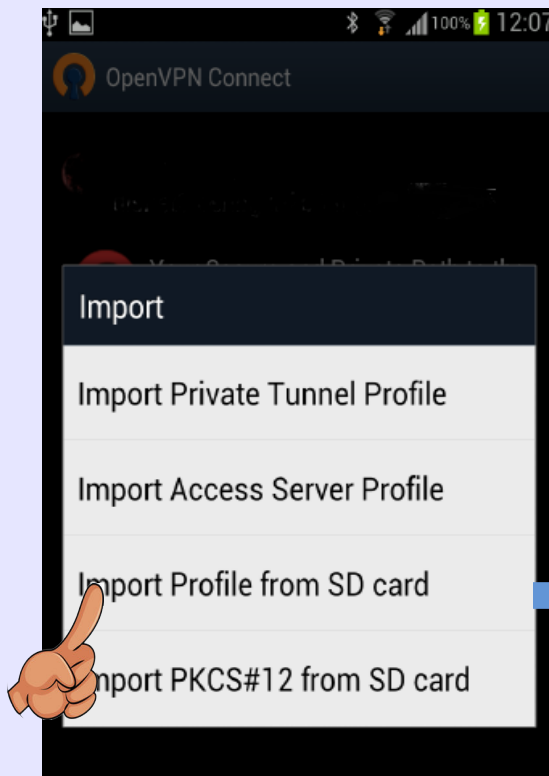
```
leCNducertif Auth-Type:= PAP, crypt-password := jsgfJlsjJA , Calling-Station-Id == "00:01:02:03:04:05"  
Framed-IP-Address = a.b.c.d
```

Accepte et renvoi une adresse IP ou rejette

# Connexion VPN depuis un mobile (OpenVpn)

## Installation/configuration côté client

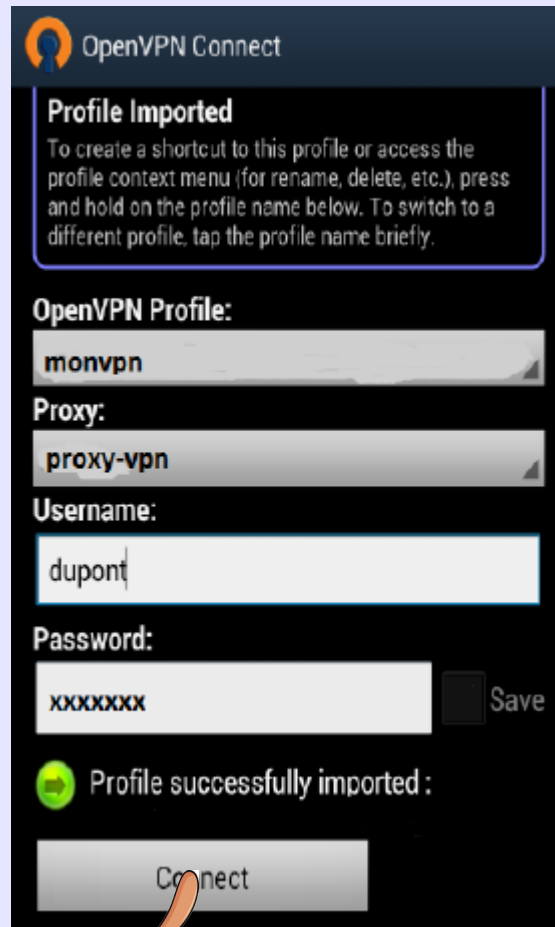
- Installer un certificat sur le mobile (comme vu précédemment)
- Installer l'application **Openvpn Connect**
- Importation de la configuration OpenVpn (profile), directement exécutée depuis une page web ou téléchargée puis importée. Une fois chargé le profile n'est plus modifiable.



```
client
verb 2
connect-retry-max 5
resolv-retry 5
dev tun
remote vpn.labo.fr 1194 udp
auth-user-pass
management-external-key
comp-lzo
setenv ALLOW_PASSWORD_SAVE 0
auth-nocache
ns-cert-type server
reneg-sec 0
```

# Connexion VPN depuis un mobile (OpenVpn)

## Connexion



OpenVPN Connect


**Profile Imported**  
To create a shortcut to this profile or access the profile context menu (for rename, delete, etc.), press and hold on the profile name below. To switch to a different profile, tap the profile name briefly.

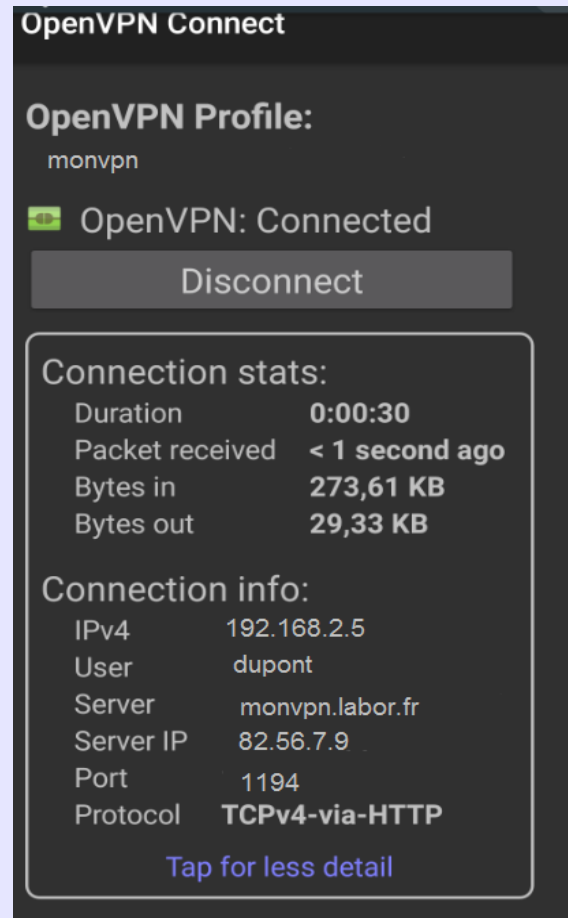
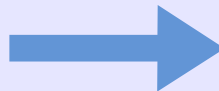
**OpenVPN Profile:**  
monvpn

**Proxy:**  
proxy-vpn

**Username:**  
dupont


**Password:**  
xxxxxxx ☐ Save

 Profile successfully imported :



OpenVPN Connect

**OpenVPN Profile:**  
monvpn

 OpenVPN: Connected

**Connection stats:**

Duration	0:00:30
Packet received	< 1 second ago
Bytes in	273,61 KB
Bytes out	29,33 KB

**Connection info:**

IPv4	192.168.2.5
User	dupont
Server	monvpn.labor.fr
Server IP	82.56.7.9
Port	1194
Protocol	TCPv4-via-HTTP

[Tap for less detail](#)

# IV

---

Sécuriser un service avec un smartphone

---

# Google authenticator

---

**Objectif** : Protéger un service en ligne par une authentification à double facteur

Le premier facteur est un mot de passe OTP fourni par un smartphone

Le deuxième facteur est un mot de passe secret de l'utilisateur

**Google Authenticator** est une implémentation de **Oath** (Open authentication)

<http://www.nongnu.org/oath-toolkit/index.html>

Avantages :

- Echec des attaques par force brute
- L'utilisateur n'a pas besoin de mémoriser un mot de passe compliqué
- Seuls les appareils qui ont une clé peuvent se connecter.
- Fonctionnement indépendant de Google (c'est une appli)



# Google authenticator

---

## Principe

- Une clé est calculée par la commande **google-authenticator**

### **Google-authenticator**

Your new secret key is: 56ECUK3X3LRZN6BT

Your verification code is 136237

Your emergency scratch codes are:

69580485

36694702

90045400

72903574

25280217

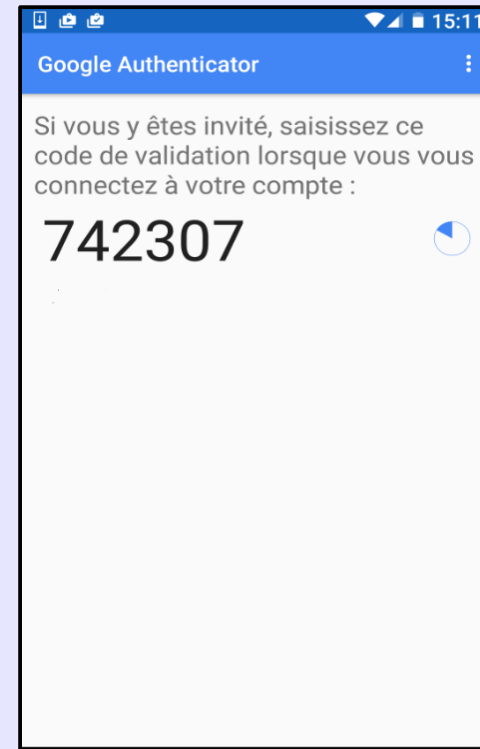
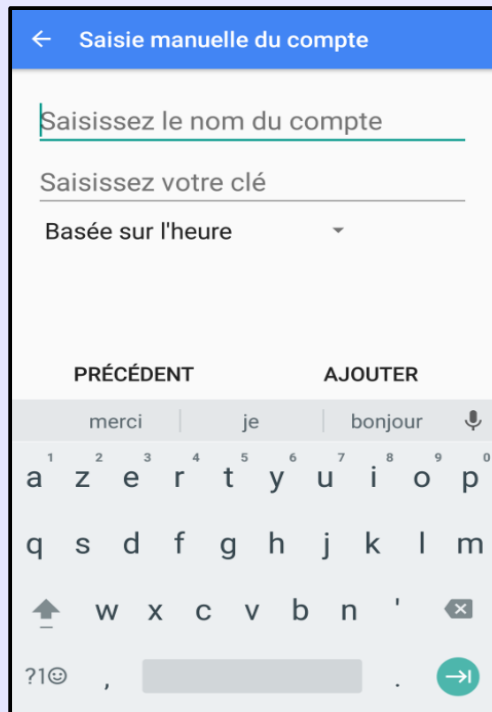
Do you want me to update your "/root/.google\_authenticator" file (y/n) y

- Cette clé est entrée sur le smartphone dans l'application **Google Authenticator**
- Cette application donnera à tout moment le code de vérification (mot de passe OTP)
- Le code de vérification peut-être testé avec la commande  
**oathtool -totp -b 56ECUK3X3LRZN6BT**



# Google authenticator

## Principe



# Google authenticator : sécuriser SSH

---

## Protéger un accès SSH des attaques force brute

### Sur le serveur SSH :

Au début du fichier `/etc/pam.d/sshd` ajouter la ligne

```
Auth required pam_google_authenticator.so
```

Dans le fichier `/etc/ssh/sshd_config`

```
ChallengeResponseAuthentication yes
```

### Dans la session utilisateur

**Google-authenticator**

Do you want authentication tokens to be time-based (y/n) **y**

Your new secret key is : **XDU5QSC66KPJZKQD**

Your verification code **158967**

Your emergency scratch codes are :

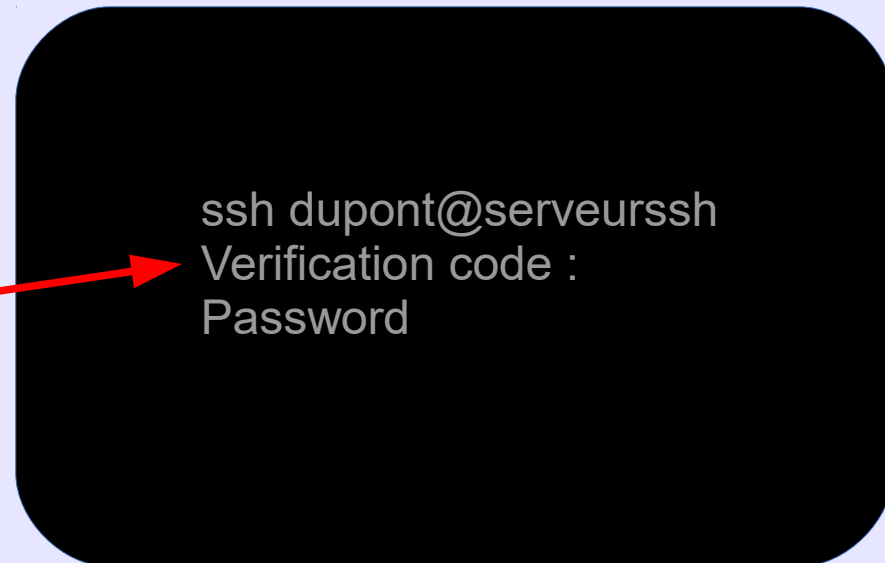
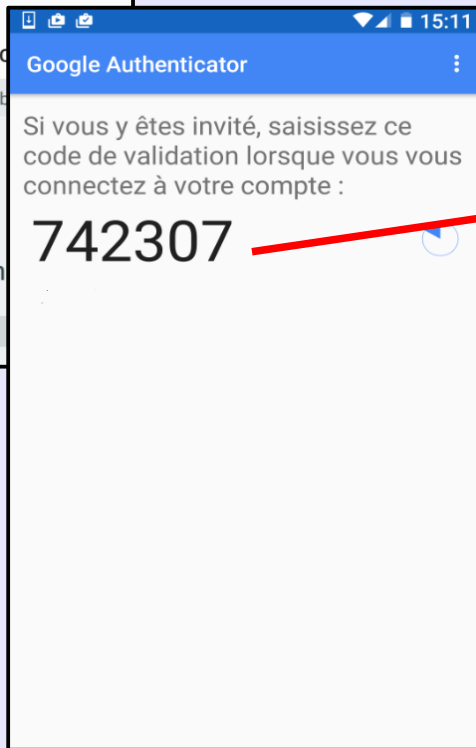
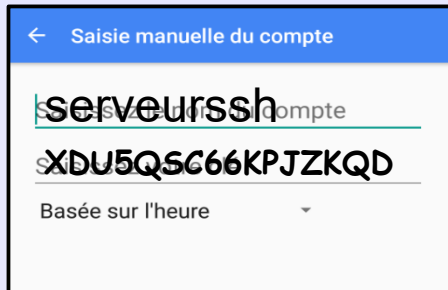
.....

Do you want me to update your « `.google_authenticator` » file (y/n) **Y**

La clé est écrite dans le fichier  
**\$HOME/ .google\_authenticator**

# Google authenticator : sécuriser SSH

## Protéger un accès SSH des attaques force brute



# Google authenticator : sécuriser OPENVPN

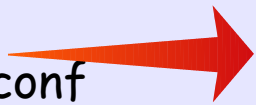
---

**Intérêt** : réaliser une authentification à double facteur sans certificat

Lors de la connexion avec le client VPN, l'utilisateur devra fournir un mot de passe statique secret suivi du code renvoyé par Google Authenticator

# Google authenticator : sécuriser OPENVPN

Du côté serveur  
Fichier openvpn.conf



```
local 192.168.0.10
port 1194
proto tcp
tmp-dir /tmp
dev tun
script-security 2
comp-lzo yes
keepalive 10 120
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so openvpn

dh /etc/openvpn/dh2048.pem
cipher AES-256-CBC
auth SHA512
server 10.8.0.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
persist-key
persist-tun
status /var/log/openvpn-status.log 20
log /var/log/openvpn.log
client-cert-not-required
tmp-dir /tmp
```

# Google authenticator : sécuriser OPENVPN

---

Du côté serveur  
/etc/pam.d/openvpn



```
auth    requisite    pam_google_authenticator.so forward_pass
```

- Simplement écrit comme ça il doit y avoir un compte utilisateur sur le serveur Openvpn
- Mais on peut aussi utiliser un authentificateur distant (ldap, NIS, radius...)

# Google authenticator : sécuriser OPENVPN

---

## ● Dans le compte utilisateur

### **Google-authenticator**

Your new secret key is: 56ECUK3X3LRZN6BT

Your verification code is 136237

Your emergency scratch codes are:

69580485

36694702

90045400

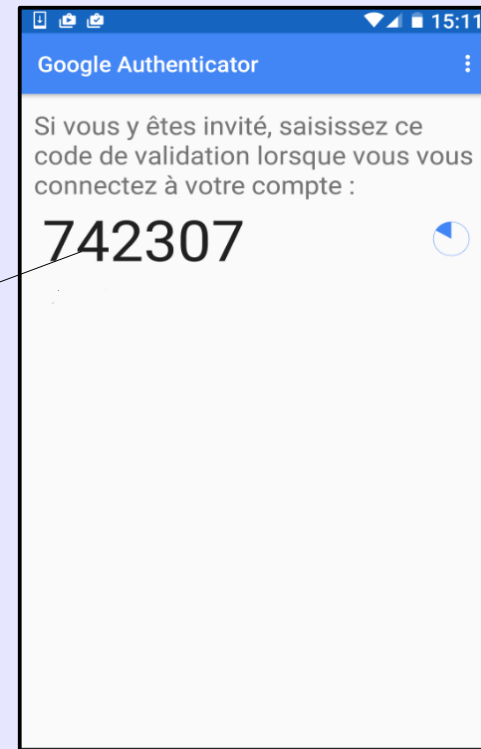
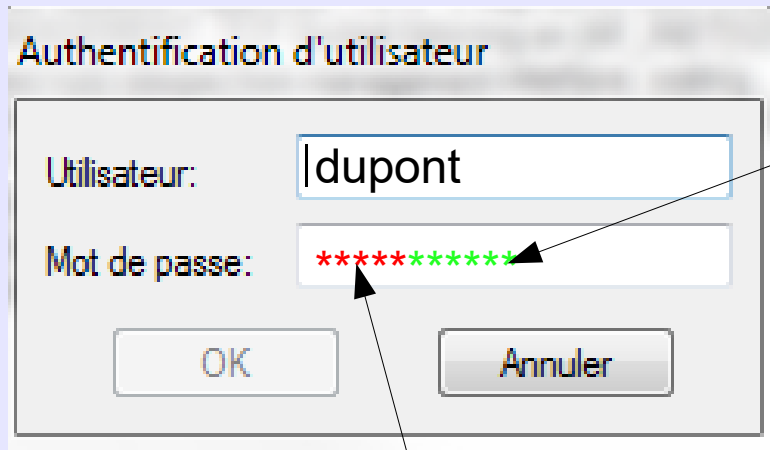
72903574

25280217

Do you want me to update your "/root/.google\_authenticator" file (y/n) y

# Google authenticator : sécuriser OPENVPN

---

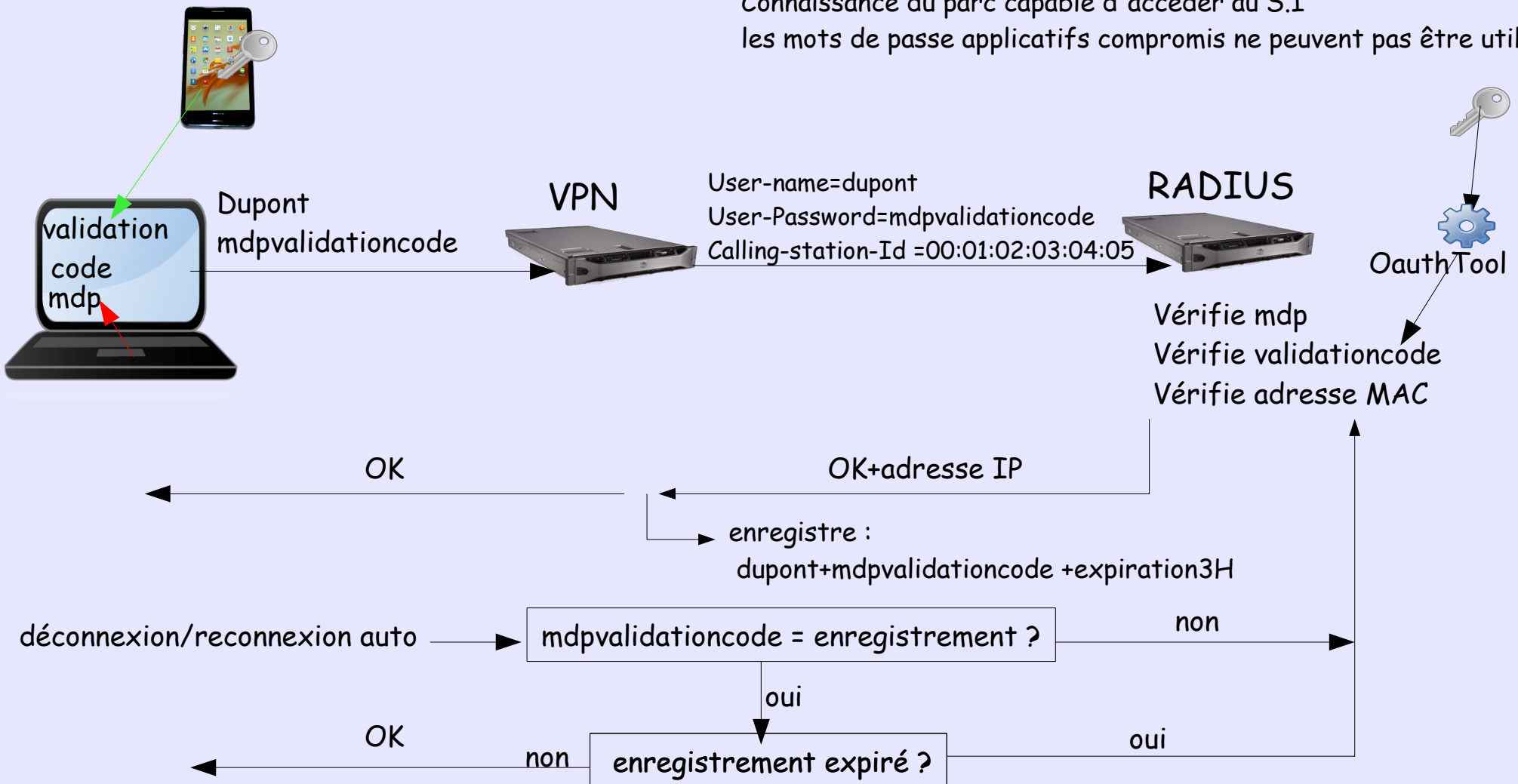


mot de passe secret



# Google authenticator : sécuriser OPENVPN

Exemple 1 de scénario ➔ Protège contre les attaques force brute  
S.I protégé même si pas de verrouillage du poste de travail  
Protège contre l'environnement privé  
Connaissance du parc capable d'accéder au S.I  
les mots de passe applicatifs compromis ne peuvent pas être utilisés



# Google authenticator

Exemple 2 de scénario : Autoriser les authentifications directes IMAP qu'après validation du mdpvalidationcode

Protège contre les attaques force brute

S.I protégé même si pas de verrouillage du poste de travail

Protège contre l'environnement privé

Pas de connaissance du parc capable d'accéder au S.I

Si le mot de passe de messagerie est compromis, il est moins facilement utilisable

