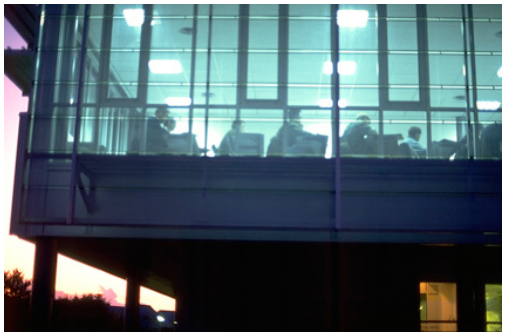


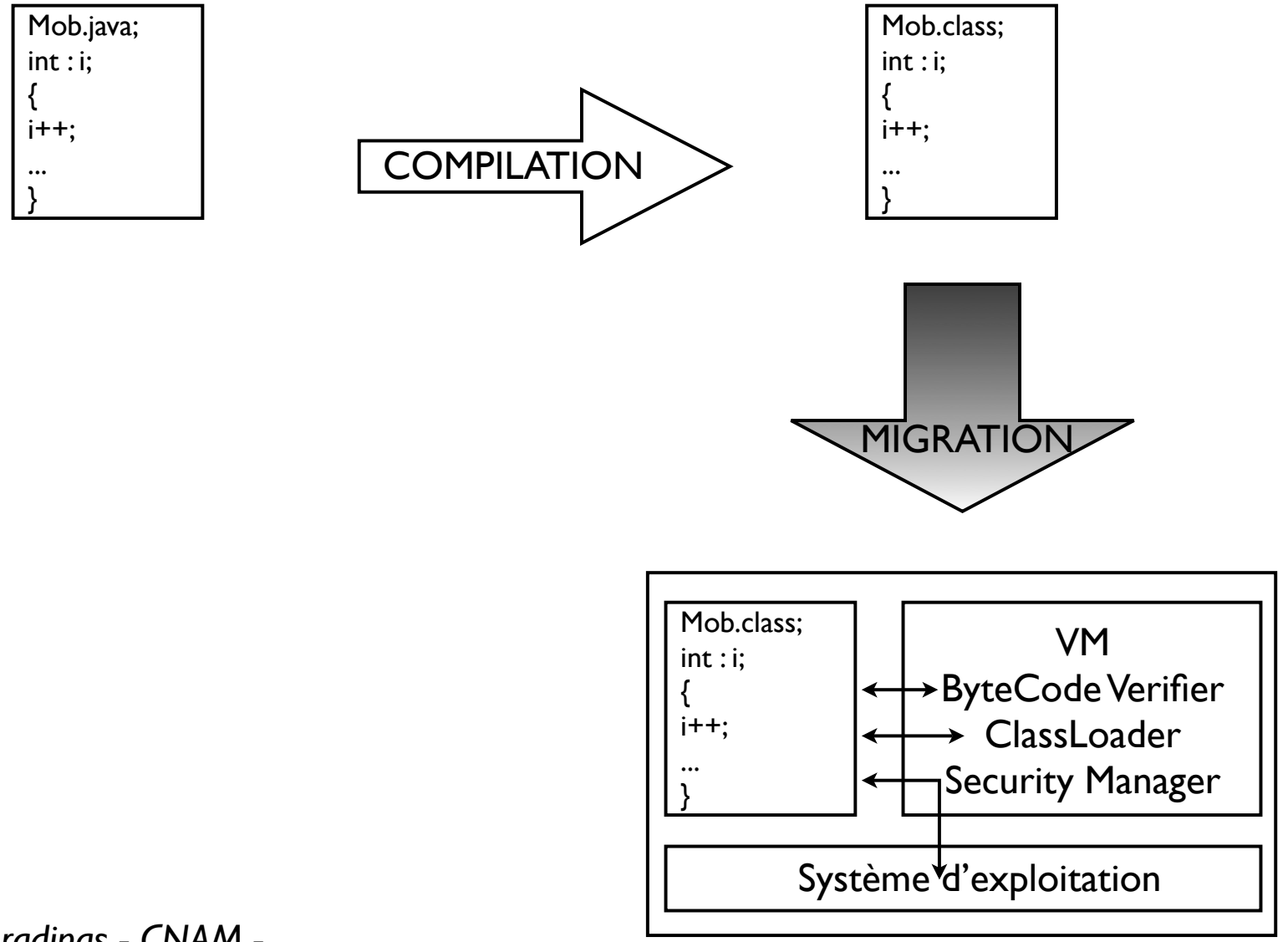
J2ME, Java Card
Architecture et Sécurité

Pierre.Paradinas / @ / cnam.fr
Cnam/Cedric
Systèmes Enfouis et Embarqués (SEE)

Java

- Mécanisme de sécurité de JavaSandbox et ses différents éléments :
  - Byte Code Verifier,
  - Class Loader,
  - Security Manager.
- Ces 3 éléments ne sont pas indépendants.

L'architecture Java



Sécurité et Java

- Aspects public des spécifications rend possible l'analyse du code,
- Le langage Java est typé et la conception est rigoureuse :
  - Les types primitifs ont des formats et tailles définis,
  - L'ordre des opérations est défini strictement,
  - Pas de pointeurs et donc d'arithmétique sur les pointeurs, de possibilité de forgé un pointeur,...
  - Les objets et méthodes sont référencés par des variables,
  - Le compilateur vérifie aussi que les références sur les objets et méthodes sont compatibles en terme de type,
  - Les références sont vérifiées (public/private,...).

Class Loader

- Java Applet Class Loader.
  - Les objets Java appartiennent tous à une classe,
  - L' Applet Class Loader détermine :
    - Quand et comment une nouvelle classe est ajoutée à l'environnement d'exécution Java en cours,
    - Que des classes ne sont pas chargées par des applets;

Le format du Class File

- Information sur le fichier,
- Un tableau contenant les informations suivantes :
  - A Unicode [6] string
  - A class or interface name
  - A reference to a field or method
  - A numeric value or a constant String value
- Les références dans le code se feront par rapport à ce tableau.
- Chaque type d'un champ ou d'une méthode est indiqué par sa signature.

Byte Code Verifier

- Le compilateur Java produit un .class, celui-ci doit être vérifié avant son exécution dans une VM.
- Le BC Verifier passera par plusieurs étapes,
- Le BC Verifier permet d'optimiser le la VM en diminuant les contrôles à l'exécution comme par exemple
  - Les accès aux registres sont valides,
  - Les paramètres des BC sont valides,
  - Les conversions de type au niveau des BC sont correctes.

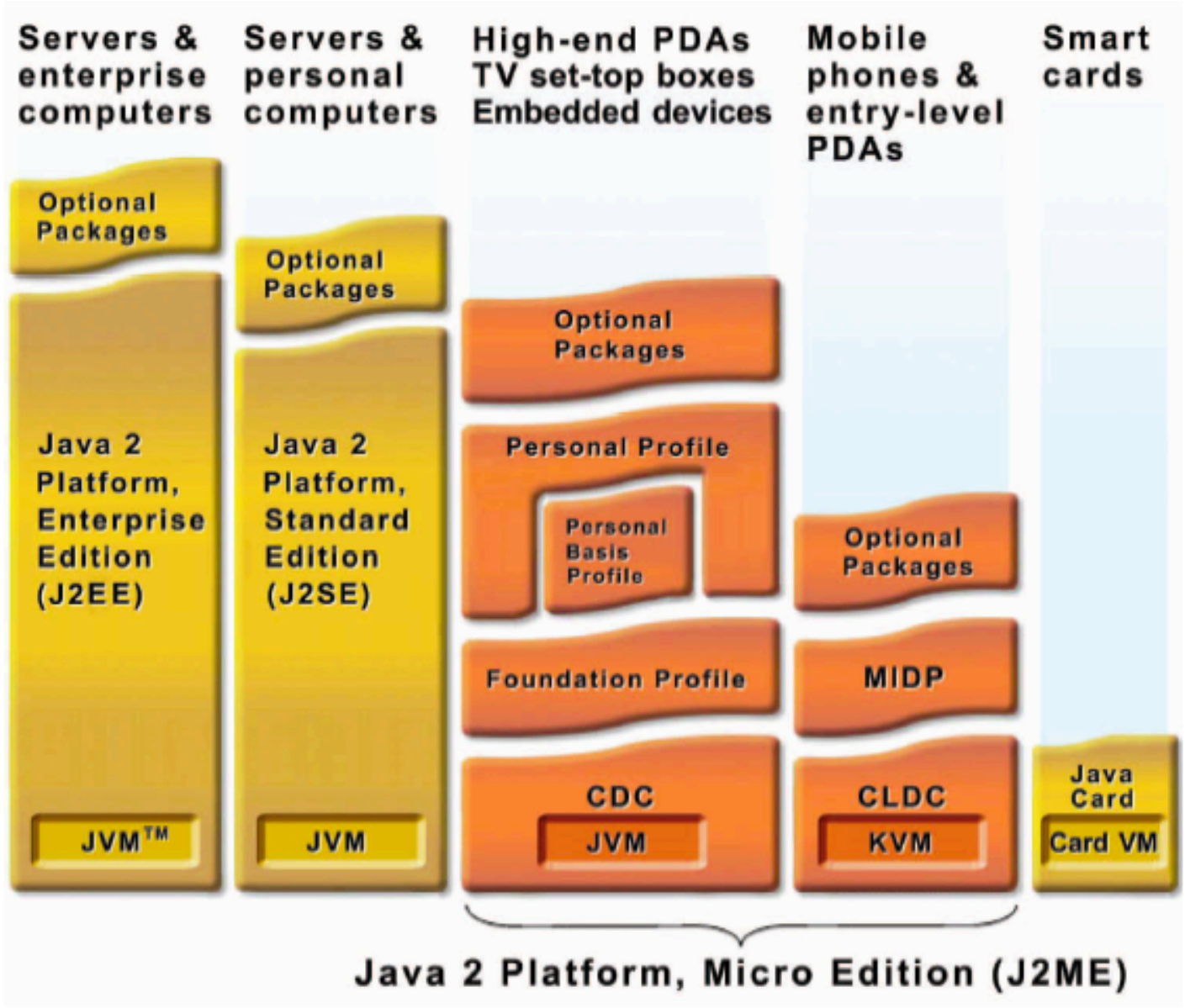
Verification

- Pass 1 :
  - Vérification du format,
- Pass 2 :
  - Vérification du format et les objest pointés existent, ... les signatures semblent être correctes

Pass 3 & 4

- Pass 3 : Phase la plus complexe
  - Analyse du Data-flow sur chaque méthode et que dans tout les cas :
    - Les méthodes sont appelées avec les bons arguments,
    - Les champs sont modifiés avec des valeurs de bon type,
    - Les opérandes ont les bons arguments sur la pile et dans les registres.
- Pass 4 :
  - Process lié au chargement de classe

Les différentes plate formes



J2ME : Java 2 Micro Edition

- J2ME Architecture
  - Configurations :
    - VM et ensemble réduit de classes
    - Offrent à un ensemble de plate-forme matérielle “similaire” (capacité processeur, capacité mémoire, caractéristiques de connections)
    - 2 configurations
      - Connected Limited Device Configuration (CLDC) :
        - 16- or 32-bit CPUs, et 128 KB à 512 KB de mémoire
      - Connected Device Configuration (CDC) :
        - 32- bit CPUs et un minimum de 2MB.

Profils

- Pour pouvoir fonctionner une configuration doit être complétée par un Profil spécifique à un environnement et ou à un domaine.
- Un profil :
  - Ensemble d'APIs de haut niveau qui définissent un cycle de vie pour les applications, interface utilisateur, ...
  - Exemple : Mobile Information Device Profile (MIDP) qui est dédié au téléphone portable.

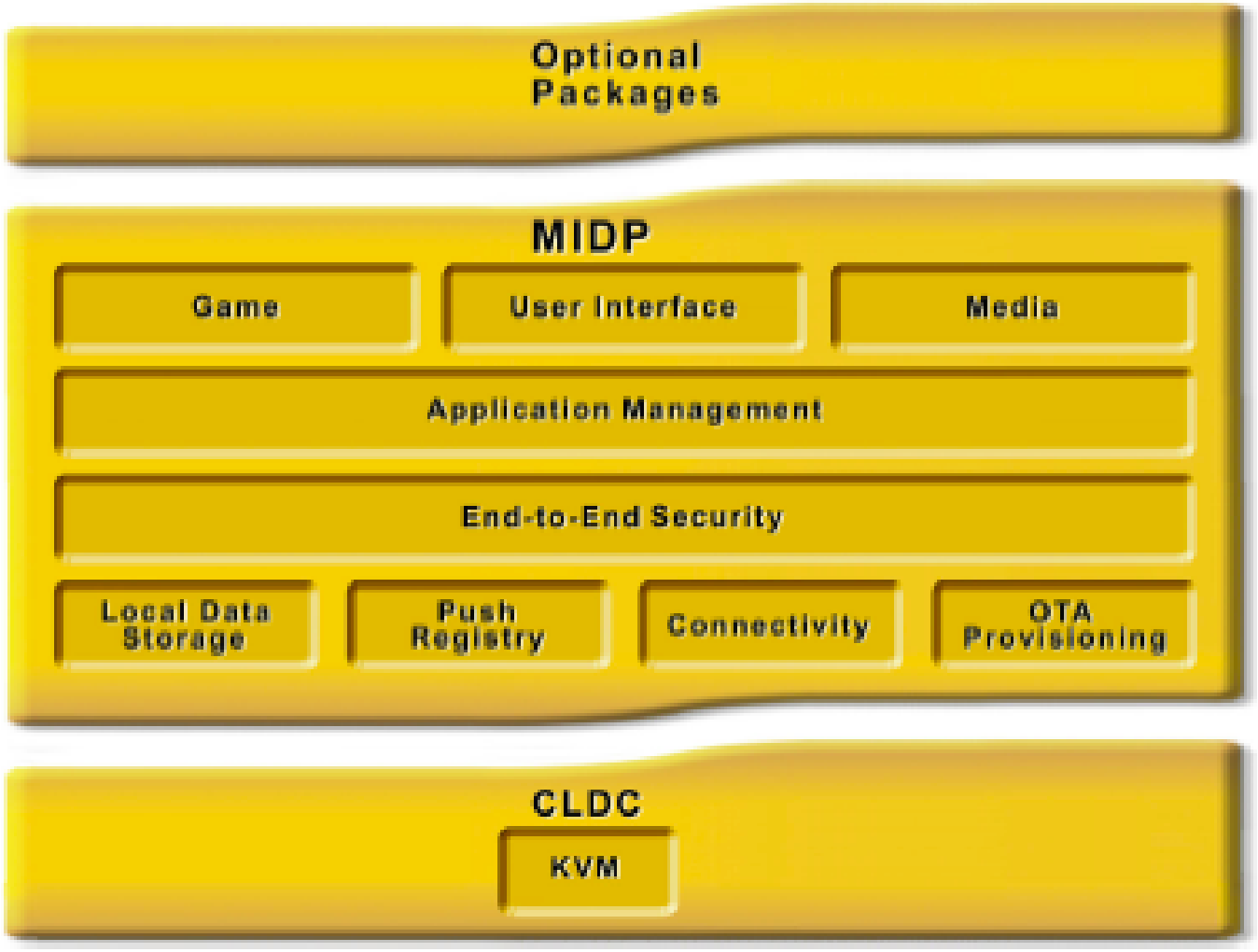
Package Optionnel

- Regroupement de fonctionalités optionnelles comme par exemple (database connectivity, wireless messaging, multimedia, Bluetooth, and web services)
- Ils ne sont pas systématiquement présents sur une plate forme.

CLDC Technologie :

- Connected Limited Device Configuration (CLDC); JSR 30, JSR 139
- Mobile Information Device Profile (MIDP); JSR 37, JSR 118
- Information Module Profile, (IMP); JSR 195
- Java Technology for the Wireless Industry, (JTWI); JSR 185
- Wireless Messaging API (WMA); JSR 120, JSR 205
- Mobile Media API (MMAPI); JSR 135
- Location API for J2ME; JSR 179
- SIP API for J2ME; JSR 180
- Security and Trust Services API for J2ME, (SATSA); JSR-177
- Mobile 3D Graphics; JSR-184
- J2ME Web Services APIs, (WSA); JSR 172
- Bluetooth API (JSR-82, Motorola, Java Partner Site)

MIDP 2.0


Optional Packages
MIDP
Game
User Interface
Media
Application Management
End-to-End Security
Local Data Storage
Push Registry
Connectivity
OTA Provisioning
CLDC
KVM

Autres aspects

- Optimized Implementations :
  - CDC HotSpot Implementation
- Java Card Technology
  - Java Card 2.2
- Comme Java, les spécifications de J2ME sont pris en charge dans le cadre du Java Community Process (http://jcp.org)

Protection proposée dans MIDP

- Le modèle applicatif :
  - Chargement de MIDlet suite est fait via le réseau (OTA),
  - Une partie des propriétés de sécurité peut être pris en charge par HTTP/WTLS
- MIDP 1.0
  - Midlet exécuté dans une SandBox,
  - Byte Code Verification,
  - Plusieurs aspects de J2SE ne sont pas présent dans MIDP :
    - Package de sécurité,
    - Fonction cryptographique,

Les évolutions de MIDP 2.0

- MIDlet de confiance :
  - Signature de la MIDlet et vérification par le terminal de la validité de ces éléments,
    - Basé sur les standards Internet, RSA et SHA-1
    - Basé sur X509 et PKI, la carte SIM/WIM ou le terminal mémorisant le “root certificate”,
  - Domaine de confiance :
    - La spécification 2.0 établit des domaines,
    - Pour les applications avec GSM et UMTS, il y a des domaines recommandés

Domaines recommandés

- 4 domaines
  - Manufacturer domains
  - Operator domains
  - Third party domains
  - Untrusted domains
- Les MIDlet d'un domaine bénéficient d'un ensemble de permission sur leur domaine.

Domaines et permissions

- Manufacturer domains
  - Toutes permissions
- Operator domains
  - Toutes permissions
- Third party domains
  - Sous contrôle de l'utlisateur
- Untrusted domains

Permissions utilisateur

- 2 types :
  - Autoriser sans avis de l'utilisateur,
  - Autoriser par l'utilisateur :
    - Toujours,
    - Par session de la MIDlet,
    - Par requête.
  - Peut être en "dur", chargé par l'opérateur ou à la charge de l'utilisateur.

Eléments de cryptographie

- Système à clé une fois
- Système à clé secrète
- Système à clé publique

La signature de code

```
MobCod ();
int : i;
{
i++;
...
}
```

Orig
(PubK, PrivK)

GENERATION

```
Mob.JAR;
int : i;
{
i++;
...
}
```

Sig (MobCod, PrivK)
Hash (MobCod)

MIGRATION

VERIFCATION {Sig (MobCod, PrivK),PubK}
VERIFCATION Hash (MobCod)

Exemple : Apple Product Security key

- Ou la migration manuelle
  - This is our PGP key which is valid until May 15, 2005
    Key ID: 0x08CF84D0
    Key Type: RSA
    Expires: 5/15/05
    Key Size: 2048/2048
    Fingerprint: ADDE FA62 D9B7 013E 07BC 8971 9CB0 E68C 08CF 84D0
    UserID: Apple Product Security

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 8.0.2

mQENBECS0jcBCAC6IwH8fWxHUKJ0bSymxiAze6WPHtJLIuDoZOfsXg0YoO68K1Il
...
cd9cIuncMyEk+FVaEet5WC8/MF3HLrADKeb748HBENOYXwXo1xZesMJiCA2yjg==
=ynQ7
-----END PGP PUBLIC KEY BLOCK-----

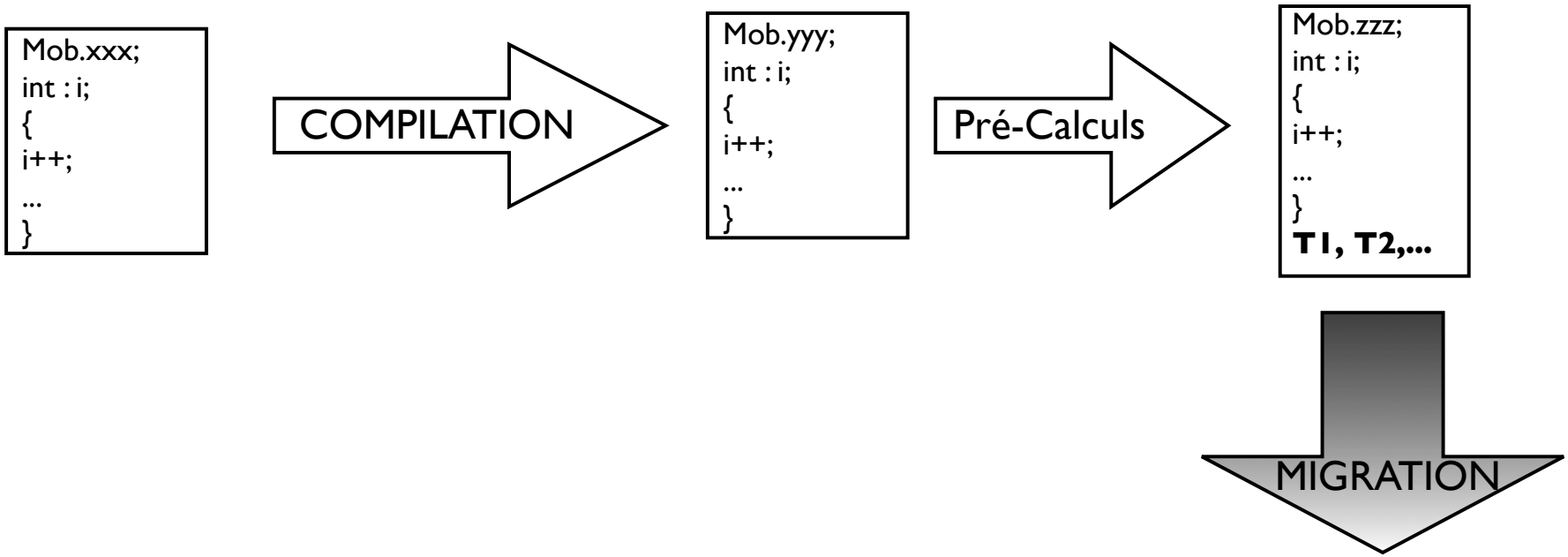
Pros/Cons

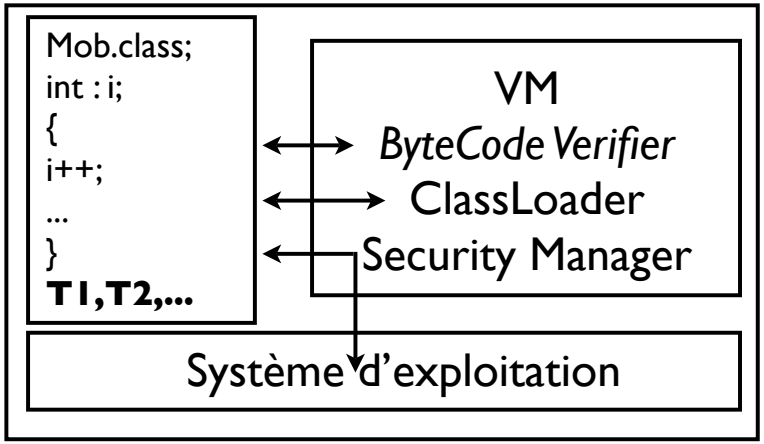
- Origine de la MIDlet est connue par la signature du code,
- De même l'intégrité peut être vérifiée à l'aide d'une fonction de hachage sur le code mobile,
- Authenticité et intégrité peuvent être assurée.
- De l'inocuité du code ?

Autre approche : PCC

- Le coût de la BC vérification est très élevé,
- L'idée du PCC (Proof Carrying Code, [Necula]) c'est de faciliter la preuve en apportant des données supplémentaires au vérifieur,
- Peut être utilisé dans les petits systèmes (carte, téléphone mobile,...)

PCC


Mob.xxx;
int : i;
{
i++;
...
}
COMPILATION
Mob.yyy;
int : i;
{
i++;
...
}
Pré-Calculs
Mob.zzz;
int : i;
{
i++;
...
}
T1, T2,...
MIGRATION


Mob.class;
int : i;
{
i++;
...
}
T1,T2,...
VM
ByteCode Verifier
ClassLoader
Security Manager
Système d'exploitation

Sur Java

- Le mécanisme de SandBox permet d'offrir une bonne sécurité
- C'est la combinaison du chargeur de classe, byte code verifier et du controleur de sécurité.
- Tant que pas de "trou" dans ce système...
  - Java Security: Hostile Applets, Holes, and Antidotes (http://www.rstcorp.com/java-security.html)
  - Princeton University's Secure Internet Programming Team (http://www.cs.princeton.edu/sip/)