Advanced Card Systems Ltd.
Card & Reader Technologies

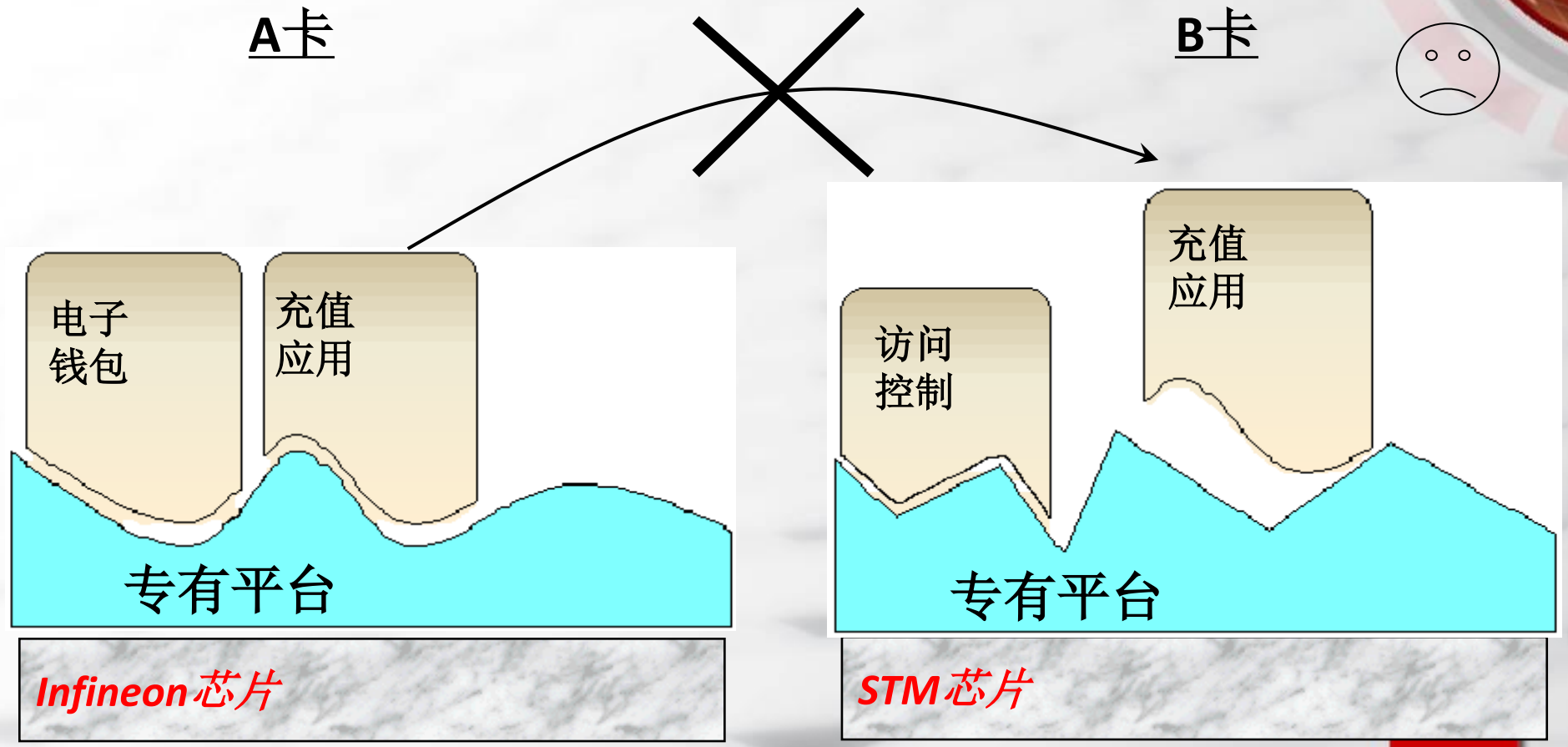
开放式平台

CARD & READER TECHNOLOGIES

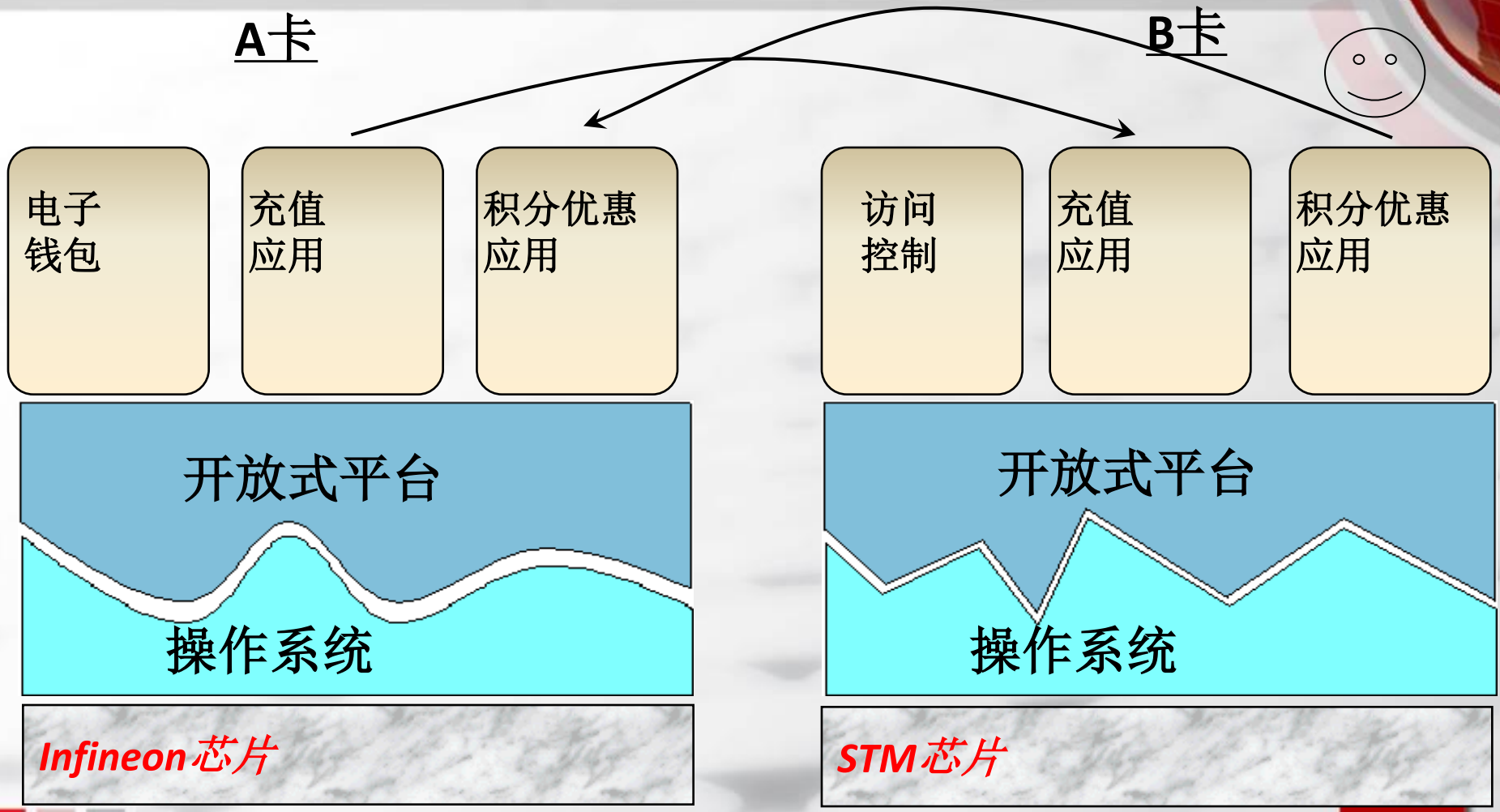
智能卡：卡片操作系统平台

- 专有（native）
  - 不能在各家供应商之间移植
  - 不对外开放
  - 单一来源
- 开放
  - MultOS卡
  - Java卡
  - 微软智能IC卡视窗

基于专有平台的不可移植的应用



基于开放平台的可移植的应用



开放式平台：卡的历史

- Mondex的概念于1990年提出，并于1993年12月正式推出。
- MultOS发明于1993年，使用Philips和Hitachi的芯片。平台由Keycorp和Dai Nippon开发（卡片生产商只需进行嵌入操作即可）。
- 同一时期，VISA的可抛弃式储值卡在多个项目中试用。
- 受塑料证卡打印机所带来的高附加值吸引，系统嵌入商开始涌现。
- 1996年，MasterCard从MultOS获得51%的股份，Mondex成为一个Applet。
- 同样在1996年，Java卡的概念被提出，并且得到Visa、Sun Microsystem和主要卡片生产商的支持。

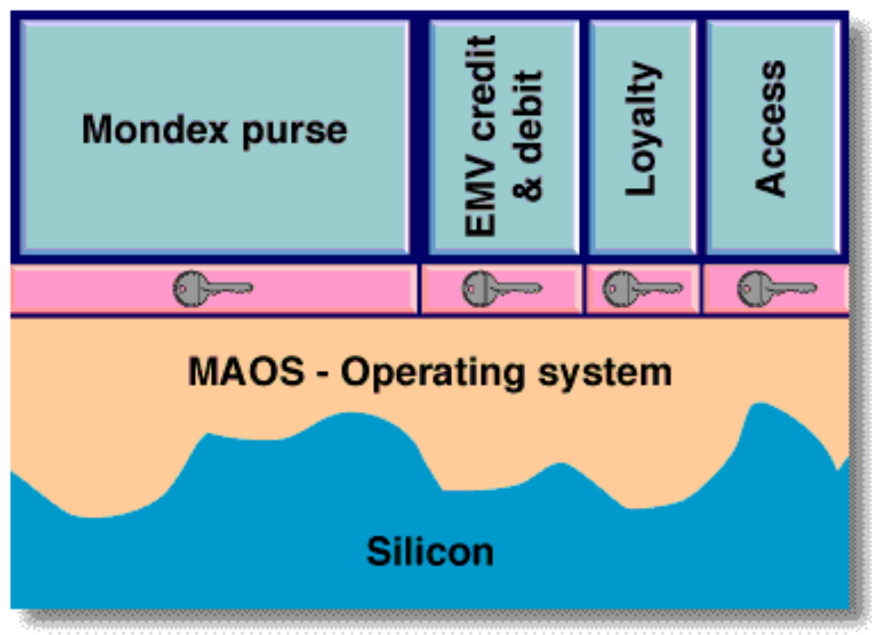
开放式平台：卡的历史

- Visa定义出Visa开放式平台（VOP），以弥补Java卡的不足。
- 1998年，GSM SIM卡工具包发行。Java顺利成章成为Applet开发的首选。
- 微软于1998年10月发布微软智能IC卡视窗，并于1999年11月正式推出。
- 1999年10月，Visa将VOP更名为全球平台(GP)并对外开放，使其成为兼容微软智能IC卡视窗和Java SIM卡工具包、并且独立于供应商的规范。
- 微软提交了源代码，并于2001年5月宣布开始提供源代码的专利许可。
- 2008年，Gemalto从Keycop收购MultOS业务。

MultOS卡

- MAOSCO定义的多应用操作系统。
- 最初采用Philips和Hitachi芯片，之后采用Infineon芯片
- 平台操作系统由Keycorp和DNP开发
- 通过ITSEC E6认证的防火墙，用于隔离卡片中的应用

MultOS卡


Mondex purse
EMV credit & debit
Loyalty
Access
MAOS - Operating system
Silicon
Applications
MULTOS
Firewalls
Silicon (hardware)
Interpreter and API
Application load certificate

MultOS卡

- 采用专用的编程语言——Multos可执行语言（MEL）
- 可以先使用C语言进行开发，之后再转换为MEL
  （比如使用Swiftcard编译器）
- 也可以使用Java开发，再编译为MEL
- MEL是一种虚拟机汇编语言

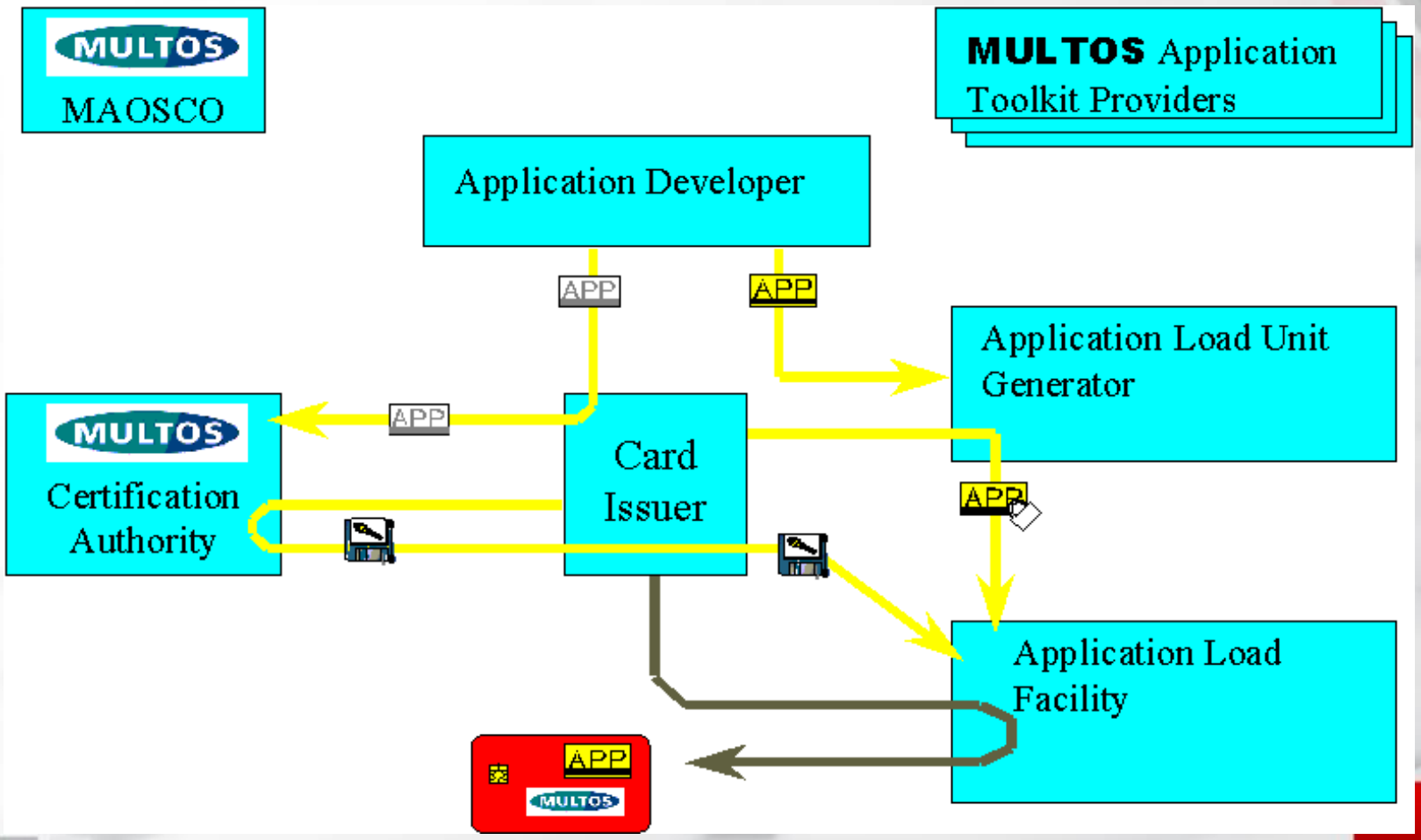
开发过程

- 使用MEL汇编语言或C编程语言在电脑上进行开发
- 在电脑上进行仿真和调试
- 与MOASCO的开发卡证书一起下载到开发卡上
  - 简单的应用开发卡
  - 高级的应用开发卡
- 应用将证书从MULTOS CA下载到用户卡

Code-let加载：充分的灵活性

- 无保护的
- 完整性保护
- 完整性&机密性保护
- 对于一张卡片或一类卡片来说，加载可以是唯一的

MultOS卡：开发过程



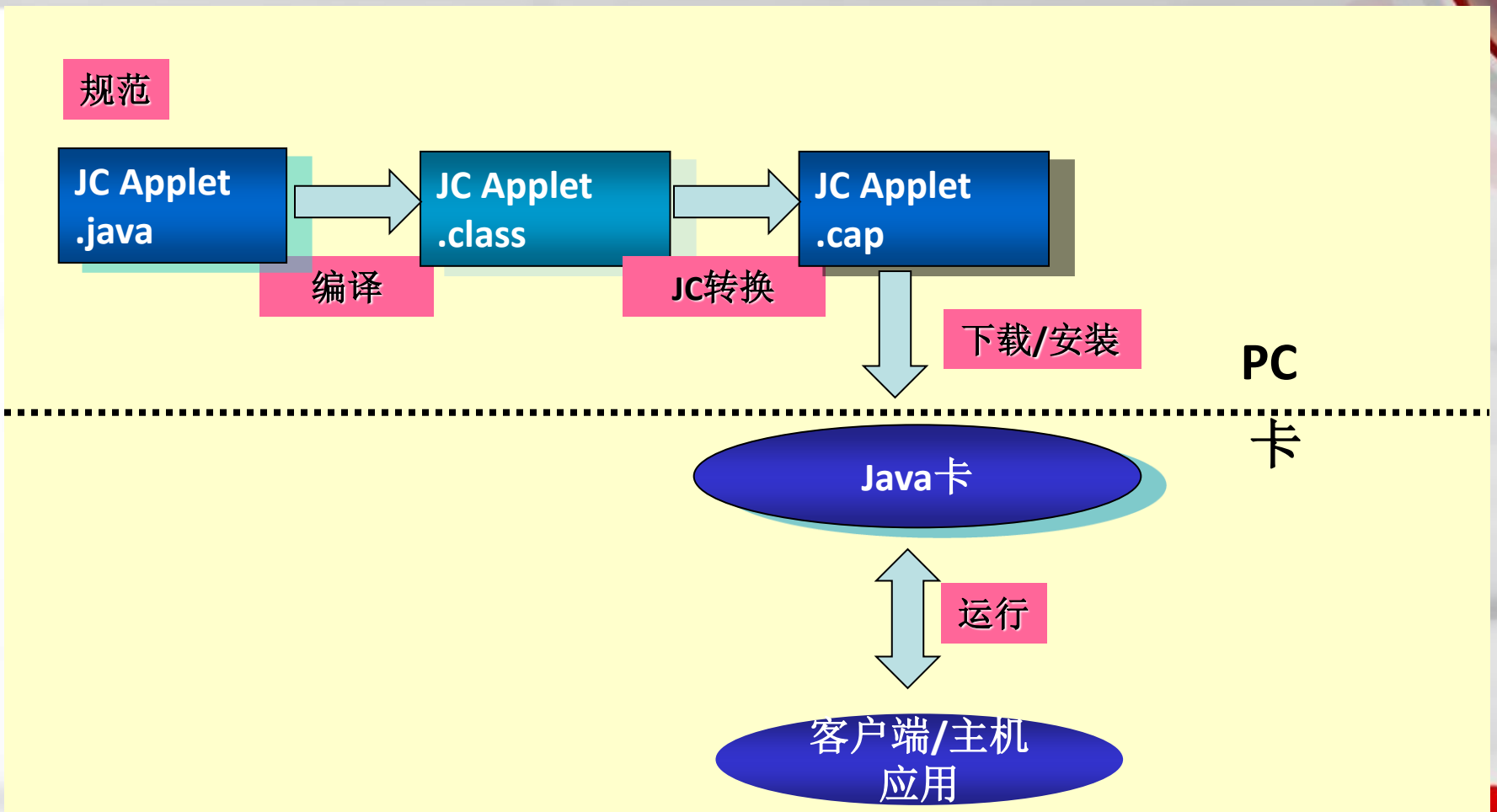
1. 许可协议。应用开发商向MAOSCO注册并购买许可证，以获得权力开发MULTOS™应用和工具。
2. 应用开发商工具包。MAOSCO采取公开应用程序开发工具的市场政策。
3. 应用开发商。MULTOS™应用程序的开发应当遵循典型的软件开发生命周期：需求、设计、编码和测试。应用程序可以直接使用MEL进行编码，也可以先使用高级语言（例如C语言）编写，再使用编译器转换为MEL语言。
4. 应用注册。作为发卡方，银行希望从所发行的每张卡片中获取最大的灵活性和利润。发卡方可以将其MULTOS™卡内的空间出售给超市等公司，以便收回卡片成本。MULTOS™ CA为MULTOS™计划提供加密服务，允许发卡方申请多个应用下载证书（ALC）来注册多个独立的应用。要注册一个应用，发卡方要从应用提供商那里获得特定的应用信息。注：无需将任何应用程序代码或数据发送给MULTOS™ CA或发卡方来进行注册。

5. 应用程序源代码。商场按照业务需求开发积分优惠应用。开发完成后，应将MEL代码、数据、ISO目录（DIR）和ISO文件控制信息（FCI）打包在一个应用下载单元（ALU）中。应用提供商、发卡方、中心甚至第三方都可以执行这一操作。

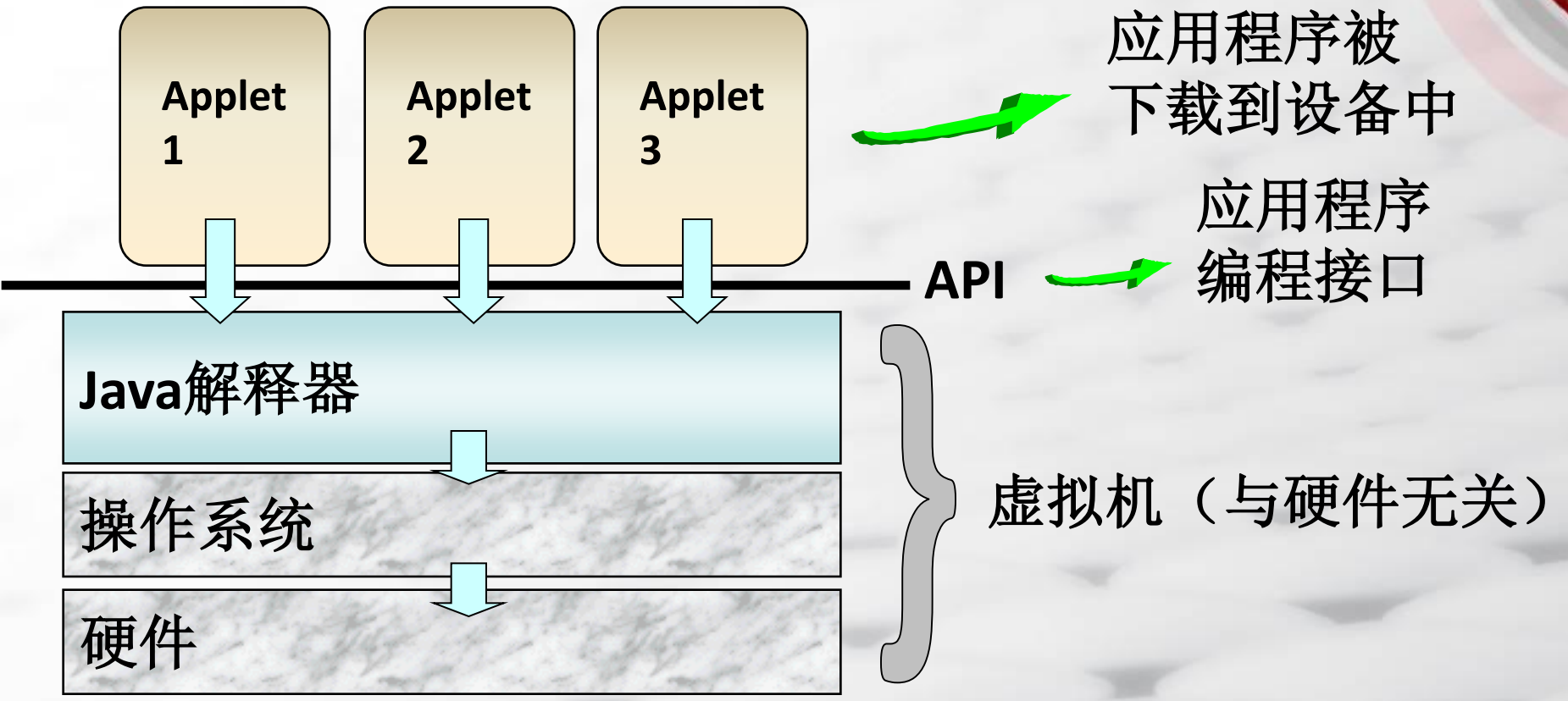
6. 应用下载单元（ALU）。ALU提供两种安全功能：
    1. 保护机密的应用程序源代码和数据。数据被加密并且只有被安全下载到正确的MULTOS卡后才会解密。
    2. 可以对ALU进行数字签名来确保数据的完整性，防止ALU发生损毁。签名由MULTOS™卡在应用下载阶段检查。

7. 应用下载证书（ALC）。ALC使发卡方能够控制哪些程序可以加载到卡片上。ALC唯一标识发卡方、应用以及应用要下载到的卡。ALC还可以包含一些其它信息，这样MULTOS™可以校验所加载应用的真实性。发卡方从MULTOS™ CA获得ALC，并将其发给应用负载设备。在该应用中，超市请求银行允许将自己的积分优惠应用加载到银行卡中。银行从MULTOS CA获得ALC，并将其发给超市。
8. 加载MULTOS™应用。用于向MULTOS™卡加载ALU和ALC的是IFD命令。应用下载证书ALC用于帮助卡片确保要下载的应用已经由发卡方批准加载到卡片中。证书由认证中心（CA）基于应用提供商给出的信息生成。传递给CA的信息要使用发卡方或应用提供商的私钥进行加密。为了检查证书的有效性，卡片需要有生成ALC的一方的公钥。公钥包含在ALC中。卡片接收到ALC后对其进行解密。

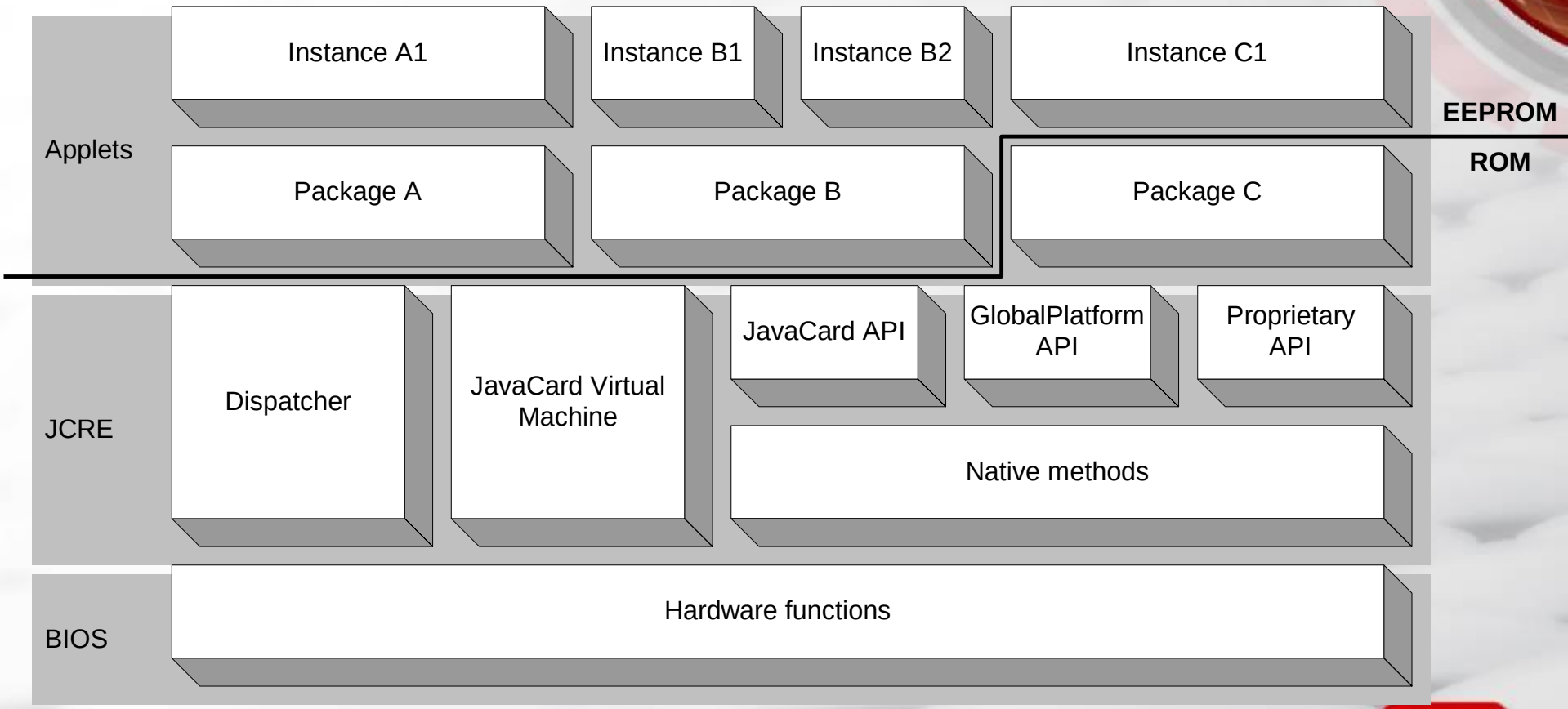
Java卡


规范
JC Applet .java
编译
JC Applet .class
JC转换
JC Applet .cap
下载/安装
PC
卡
Java卡
运行
客户端/主机应用

Java卡


Applet 1
Applet 2
Applet 3
应用程序被下载到设备中
API
应用程序编程接口
Java解释器
操作系统
硬件
虚拟机（与硬件无关）

Java卡架构


Applets
Instance A1
Instance B1
Instance B2
Instance C1
Package A
Package B
Package C
EEPROM
ROM
JCRE
Dispatcher
JavaCard Virtual Machine
JavaCard API
GlobalPlatform API
Proprietary API
Native methods
BIOS
Hardware functions

Java卡介绍

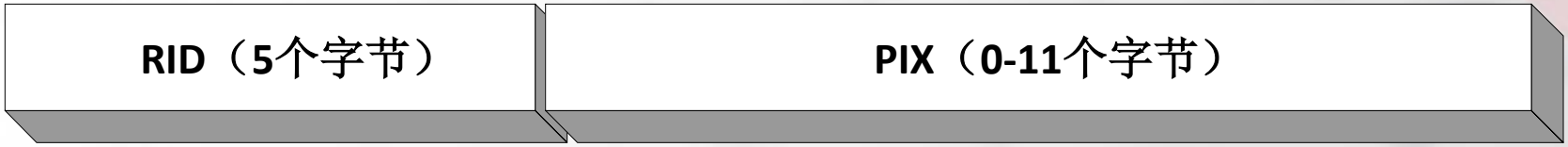
- Java卡允许：
  - 在一张卡片上可以同时存在多个独立的应用
  - 发卡后可以重新添加新的应用
  - 应用具有源代码和二进制兼容性
- Java卡定义：
  - 一种语言
  - 一种标准二进制格式的Applet（Bytecode）
  - 一种虚拟机，用于执行Applet Bytecode
  - 一组供Applet调用的API
  - Applet选择机制
  - 无文件系统

Java卡Applet

- 包
  - 可以包含多个Applet类
  - 对应Applet下载时传输给卡片的数据单元
- Applet类
  - 包含Applet代码
- Applet实例
  - 包含Applet数据
  - 同一Applet类可以有多个实例

AID：应用标识符

- 唯一地标识Applet类、实例和包



- RID：应用提供商的唯一标识符
  （由ISO提供；例如A0 00 00 00 77）
- PIX：指定RID下的PIX是唯一的，用于标识应用（由应用提供商分配）

Java卡运行时的环境

- Java卡虚拟机
  - 解释（运行）Applet代码（bytecode）
  - Bytecode是一种抽象的机器语言，独立于任何芯片硬件
- Java卡API
  - 为Applet提供多种服务：APDU命令管理、加密等等
- 分配器
  - 负责Applet选择机制，将命令传递给合适的Applet

Applet选择

- 分配器接收来自终端的所有APDU命令。
- 处理SELECT APPLET命令，并记录当前所选的Applet
- SELECT APPLET命令在数据域中给出要选择的Applet实例的AID。
- 分配器将之后收到的所有命令发送给当前所选的Applet。
- 上电时选择默认的Applet。

Java卡语言子集

- 支持的Java特性
  - 小的原始数据类型：boolean、byte和short
  - 一维数组
  - Java包、类、接口
  - 继承、虚方法、重载
  - 动态对象创建
  - 类成员访问范围
  - 异常处理
  - 垃圾回收（JC 2.2）
- **不支持的Java特性**
  - **大的原始数据类型：int（可选）、long、double和float**
  - **char类型和strings**
  - **多维数组**
  - **线程**
  - **动态类下载**
  - **沙箱模型和安全管理器**
  - **内省**
  - **对象序列化**
  - **对象复制**

Java卡API

- 标准的Java API并不适合智能卡。
- Java卡最低标准的Java类包括：
  - *Object*类
  - *Throwable*类和一些基本的异常类
  - 无其它标准Java包
- Java卡提供三个专门用于智能卡的包：
  - *javacard.framework*
  - *javacard.security*
  - *javacardx.crypto*

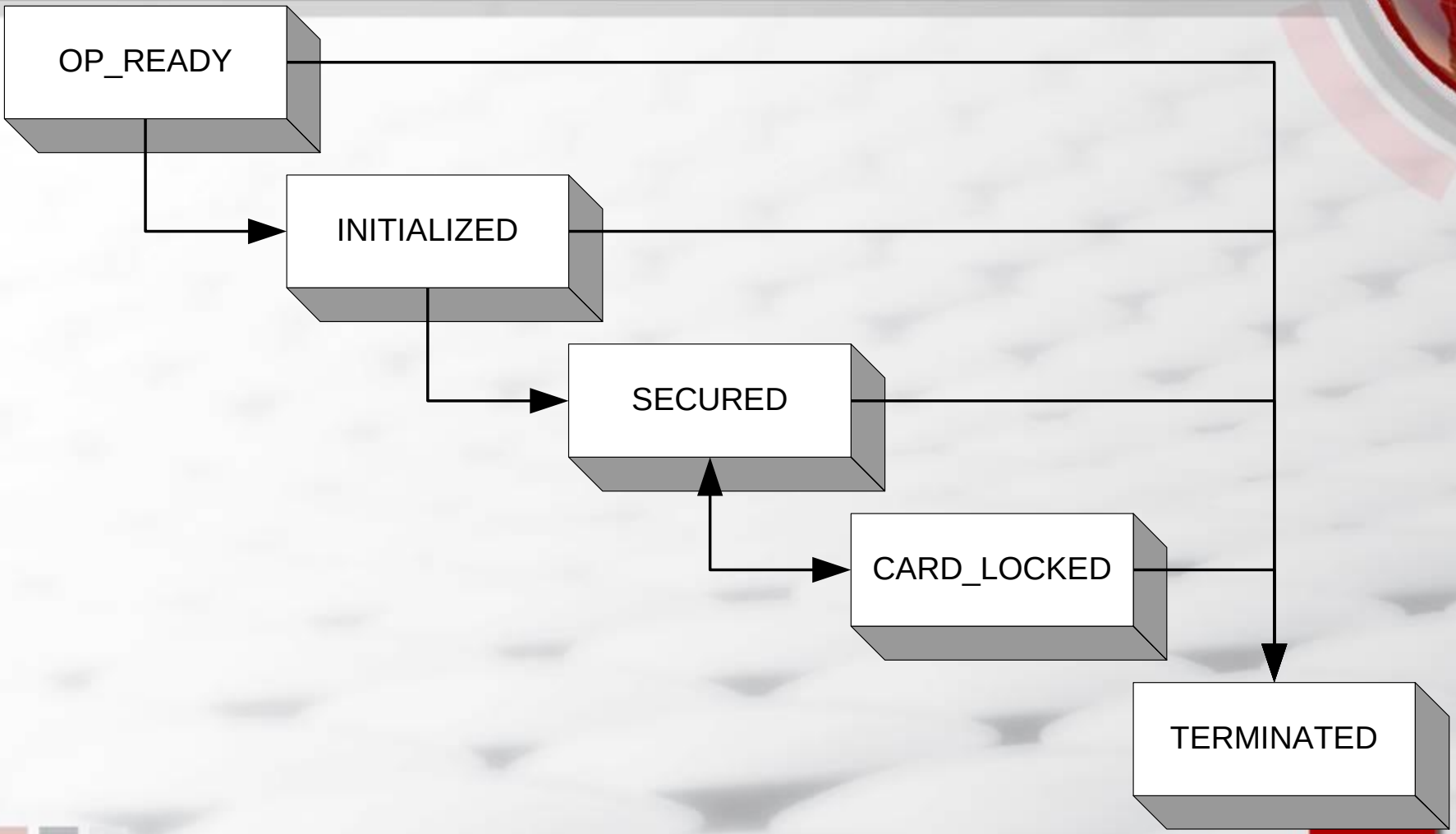
全球平台（GlobalPlatform）介绍

- 之前称为Visa开放平台（VOP）
- GlobalPlatform允许：
  - 以一种安全的方式与卡片进行通信
  - Applet下载、安装和删除
  - 为每个应用提供商定义特定的安全环境
- GlobalPlatform定义
  - 卡片管理器应用
  - 安全域应用
  - 所有其他应用的通用操作
  - API

卡片管理器（Card Manager）

- 卡片管理器是一个Applet，
- 又称为发卡方安全域（Issuer Security Domain）。
- 在卡片上安装的第一个Applet。
- 通常在卡片上电时默认选择。
- 是卡片内容管理的入口点。
- 其APDU命令由GP制定。

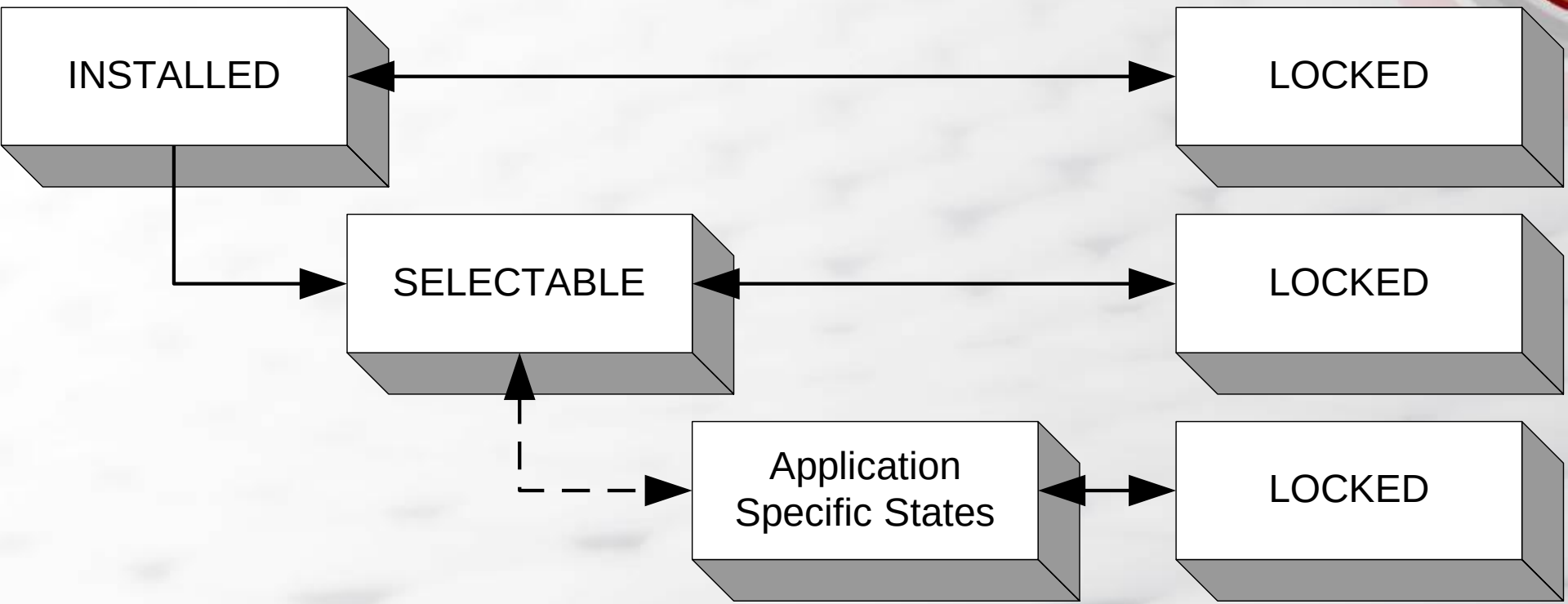
卡片管理器生命周期



GP APDU命令

- Delete
- Get Data
- Get Status
- Install
- Load
- Manage Channel
- Put Key
- Select
- Store Data
- Initialize Update
- External Authenticate
- Begin R-MAC Session
- End R-MAC Session
- Get Challenge
- Internal Authenticate
- Manage Security Environment
- Perform Security Operation

应用生命周期


INSTALLED
LOCKED
SELECTABLE
LOCKED
Application
Specific States
LOCKED

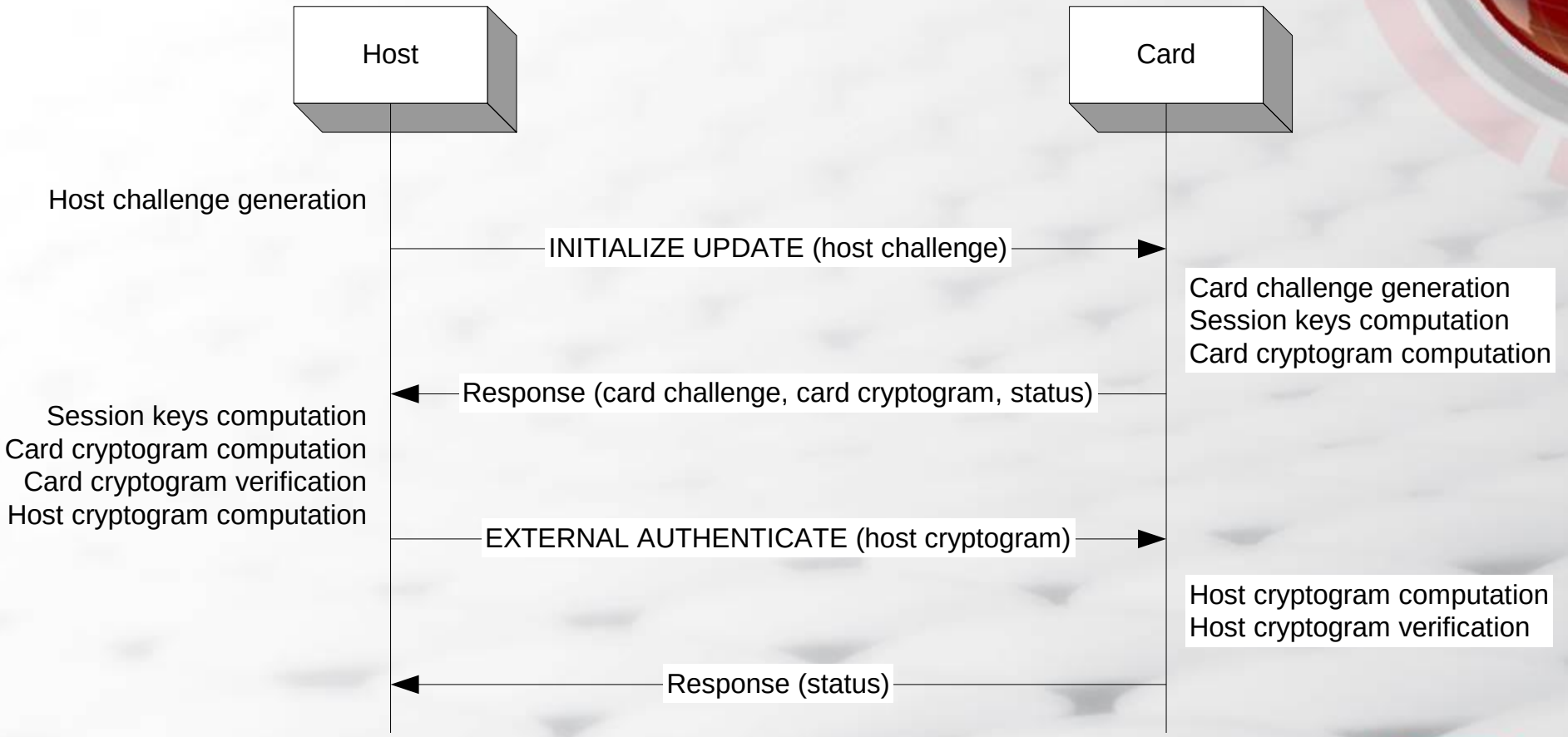
安全报文发送

- 卡和主机之间的安全信道
- 卡片管理器进行的敏感操作都要用到安全报文
- 任何一个应用都可以采用的机制
- 多个安全通道协议（在预个人化期间定义）
  - SCP01模式5：与OP2.0.1’兼容性
  - SCP02模式55：提高了安全性
- 三个安全级别（SC开启时选择）
  - 相互认证
  - 相互认证和完整性检查
  - 相互认证、完整性检查和机密性

密钥集

- 卡片管理器和安全域都可以容纳多个密钥集。
- 密钥集靠版本号来识别。
- 要开启安全通道，必须了解密钥集的相关内容
- 一个密钥集由三个密钥组成：
  - 加密/认证密钥
  - MAC密钥
  - 密钥加密密钥

开启安全通道


Host
Card
Host challenge generation
INITIALIZE UPDATE (host challenge)
Card challenge generation
Session keys computation
Card cryptogram computation
Response (card challenge, card cryptogram, status)
Session keys computation
Card cryptogram computation
Card cryptogram verification
Host cryptogram computation
EXTERNAL AUTHENTICATE (host cryptogram)
Host cryptogram computation
Host cryptogram verification
Response (status)

包的下载

- INSTALL for LOAD命令
  - 要下载的包的AID
  - 与包相关的安全域的AID（或者卡片管理器的AID（如适用））
- LOAD命令
  - 发送加载文件的一部分给卡片
  - 加载文件中包含了CAP文件，封装在TLV内
  - 加载文件内还可能包含了数据鉴别块（DAP block，如适用）

应用的安装

- INSTALL for INSTALL命令
  - 含有applet的包的AID
  - 包中的applet类的AID
  - 必须分配给实例的AID
  - 应用的权限
  - 安装参数
- 通常与INSTALL for the MAKE SELECTABLE命令相结合

安全域（Security Domain）

- 安全域是一个applet。
- 它代表了应用提供商。
- 有自己的密钥集
- 安全域可以有不同的权限
  - 简单的安全域
  - DAP（数据鉴别模式）验证和强制DAP验证
  - 带委托管理的安全域

简单的安全域

- 提供新的密钥集给需要安全报文发送服务的应用
- 这些密钥集不会授权修改卡片的内容，而卡片管理器密钥集可以
- 包在加载时会与一个SD关联起来，包中所有的applet都与其联系在一起。
- 要想修改连接并将其与另外一个SD相关联，可以释放applet。

DAP验证

- 目的
  - 检查包的代码的完整性
  - 检查包是否由经过授权的一方（应用提供商）提供
- 这些安全域需要一个额外的密钥：DAP密钥（RSA公钥）。
- DAP是对CAP文件的签名，通过DAP密钥生成。
- 包在下载时，卡片管理器要查询SD以检查DAP的值。
- DAP验证是可选的（除非包必须与SD关联），但强制DAP验证是必选的。

卡片相关应用程序的设计
采用开放平台COS

- 先了解智能卡应用的安全要求
- 设计应用的APDU命令集、应用数据和密钥
- 设计SAM的APDU命令集、应用数据和密钥
- 为所有的子系统设计SAM-应用-卡APDU交易流
- 提供测试卡并与各子系统供应商测试SAM卡，使子系统商户了解如何使用卡片和SAM。

其它问题？