La performance des plateformes Java Card

Julien Cordry

CNAM-Cédric
292 rue Saint-Martin
75003 Paris
Julien.Cordry@cnam.fr

1. Introduction

Avec plus d'un milliard de cartes vendues chaque année, les cartes à puce sont un outil important dans notre vie quotidienne. Le déploiement du standard Java Card [1] a rendu plus accessible la carte du point de vue du développement : la possibilité d'utiliser un sous ensemble du très populaire langage Java pour produire un programme qui soit indépendant de la plate-forme cible raccourcit le temps nécessaire à la commercialisation d'une application pour une carte à puce [3]. De plus, les cartes sont aujourd'hui des «plates-formes ouvertes » dans le sens où les programmes (applets) peuvent être ajoutés, c'est à dire chargés et exécutés sur ces plates-formes, par opposition à un système dédié.

Dans ce contexte, il est important pour la communauté Java Card d'analyser les performances de ces plates-formes. En dehors de notre projet, il n'y a pas de solution sur le marché qui se fixe pour but d'évaluer les performances d'une carte à puce qui implémente la technologie Java Card. Les produits qui réalisent ce type d'évaluation sont généralement propriétaires et développés dans les entreprises du domaine, ou chez les principaux clients de cette technologie, sans que l'ensemble de la communauté Java Card n'y ait accès. Toutefois, quelques initiatives académiques ont vu le jour récemment, sans que les projets n'aboutissent vraiment (le projet SCCB du laboratoire Cédric [6] ou au IBM Research Lab de Zürich [11] par exemple ne sont plus disponibles, d'autres [2, 7, 9] restent très partiaux et ne s'intéressent qu'à un sous ensemble de benchmarks [1]). Dans les travaux non publiés entrepris par des entreprises, on peut trouver ceux d'Erdmann [4] qui détaillent une méthodologie pour réaliser la mesure de performance et ceux de Fischer [5] qui comparent la performance sur certaines cartes en Java Card et en code natif, les deux travaux restant confinés à des cartes Infineon.

[1] des outils de mesure de la performance

Un benchmark de référence est utile à la communauté puisqu'il permet de discriminer les produits sur une technologie standardisée comme c'est le cas ici. De plus, être capable de mesurer avec précision la performance en termes de mémoire, de consommation énergétique et en temps d'exécution d'un outil cryptographique comme une carte à puce peut permettre d'effectuer des attaques et des évaluations de sécurité.

Le but du projet ANR Mesure [8] est de pallier à ce manque. Dans un premier temps, nous identifions les problèmes de mesures de performance dans le contexte des plates-formes Java Card. Nous proposons ensuite une méthodologie qui effectue des mesures et qui permet de fixer une note globale pour un ensemble de performances sur une carte donnée, afin de caractériser l'efficacité de la plateforme Java Card embarquée. Ainsi, nous espérons définir un standard auquel n'importe quelle carte puisse se comparer. Nos travaux sont publiés sous la forme d'une famille d'outils libres.

2. Mesure

L'essence du projet consiste à effectuer des mesures de temps d'exécution pendant qu'on envoie des APDUs de la machine hôte à la carte à travers un Card Acceptance Device (CAD, appelé aussi lecteur). Ce système vient de la nature même de l'architecture ISO 7816. D'un point de vue pratique, nous pouvons lancer un chronomètre avant d'envoyer nos APDUs à la carte. Celle-ci va traiter la commande envoyée puis répondre. Lors de la réception de la réponse de la carte par la machine hôte, celle-ci peut arrêter le chronomètre et noter le temps d'exécution. Chaque mesure est effectuée un certain nombre de fois. Le temps ainsi mesuré contient beaucoup de bruit provenant de la machine hôte ainsi que de la carte. Nous utilisons un moyen d'affiner nos mesures pour déduire le temps d'exécution d'un bytecode seul ou d'une entrée d'API seule.

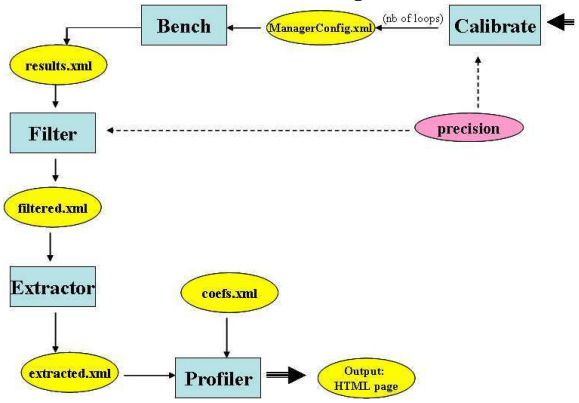
La figure 1 illustre un ensemble de modules qui forment un cadre pour nos mesures.

Il y a deux parties dans notre approche : une première partie composée de deux modules qui se chargent de faire des mesures sur la carte et une partie composée de trois parties qui se chargent de traiter les résultats recueillis.

La partie mesure à proprement parler (dans chacun des deux premiers modules) est composée d'une partie scripts de test sur la machine hôte et d'une partie applets chargées successivement sur la carte.

Chaque test effectué sur la carte consiste en une boucle de taille transmise avec l'APDU qui lance le test. Chaque tour de boucle effectue une méthode `run` de l'applet. Cette méthode effectue l'opération qui nous intéresse ainsi que des opérations qui sont nécessaires à l'exécution de celle-ci. Ceci génère du bruit côté carte qu'il faudra déduire par la suite (voir [10]).
Le premier module détermine donc la taille optimale de la boucle interne pour avoir des mesures significatives, et ce, sur des familles de tests. Le second module effectue les mesures à proprement parler sur tous les tests de chaque famille.

FIG. 1 – Architecture générale



Les modules de la partie traitement des résultats font trois choses successives :
- Un premier module écarte les mesures qui tombent trop loin de la moyenne. Une analyse de la distribution permet en effet de se rendre compte que quelques mesures font grossir l'écart type.
- Un second module fait les déductions d'opérations élémentaires : bytecodes et entrées d'API à partir des mesures brutes.
- Un module assigne une note pour chaque carte et pour chaque domaine d'utilisation (transports, bancaire, identité) ainsi qu'une note globale. Cette note correspond aux performances de la carte avec des coefficients qui viennent de l'usage fait dans chaque domaine de chaque bytecode et entrée d'API.

3. Conclusion

Les outils sont à l'heure actuelle disponibles en ligne [8], et sont publiés comme des outils libres. Les résultats sont peu sensibles aux paramètres environnementaux tels que le système d'exploitation ou la qualité du CAD. Notre travail s'est jusqu'ici concentré sur le temps d'exécution des bytecodes et des entrées d'API. D'autres mécanismes, tels que les entrées/sorties et la consommation électrique sont des problèmes ouverts. Les travaux en cours incluent la modélisation des performances ainsi que la validation de ce modèle.

Bibliographie

1. Java Card 2.2.2 Specification, April 2006 `http://java.sun.com/products/javacard/`
2. Jordi Castellà-Roca and Josep Domingo-Ferrer and Jordi Herrera-Joancomartí and Jordi Planes, A Performance Comparison of Java Cards for Micropayment Implementation, CARDIS, pages 19-38, 2000
3. Zhiqun Chen, Java Card Technology for Smart Cards : Architecture and Programmer's Guide, Addison Wesley 2000
4. Monika Erdmann, Benchmarking von Java Card, LudwigMaximilians-Universität München, Institut für Informatik, Mai 2004
5. Mario Fischer, Vergleich von Java und Native-Chipkarten Toolchains, Benchmarking, Messumgebung., Ludwig Maximilians-Universität München, Institut für Informatik, 2006
6. Gilles Grimaud and Pierre Paradinas and Eric Vétillard, Measuring the performance of the Java Card Platform, Java One, Mai 2006
7. Constantinos Markantonakis, Is the performance of smart card cryptographic functions the real bottleneck ?, 16th international conference on Information security : Trusted information : the new decade challenge, pages 77 - 91, volume 193, Kluwer, 2001
8. MESURE, The MESURE project website, `http://mesure.gforge.inria.fr/Eng/Index`
9. Konstantinos Papapanagiotou and Constantinos Markantonakis and Qing Zhang and William G. Sirett and Keith Mayes, On the Performance of Certificate Revocation Protocols Based on a Java Card Certificate Client Implementation, 20th IFIP International Information Security Conference (Sec 2005) - Small Systems Security and Smart cards, Mai 2005
10. Pierre Paradinas and Julien Cordry and Samia Bouzefrane, Performance Evaluation of Java Card Bytecodes, WISTP, pages 127-137, Mai 2007
11. Karima Rehioui, Java Card Performance Test Framework, Université de Nice, Sophia-Antipolis, IBM Research internship, Septembre 2005