Transactions mobiles & sécurité des smartphones

Jean Claude Paillès
Université de Caen Basse-Normandie, UMR 6072 GREYC, F-14032 Caen, France
ENSICAEN, UMR 6072 GREYC, F-14050, Caen, France
Email: jc.pailles@voila.fr;

Résumé: les smartphones vont très bientôt permettre de gérer les transactions de la vie quotidienne (paiement, ticketing, etc) à distance (via le Net) ou en face à face, grâce au NFC. Ces transactions sont clairement sensibles sur le plan de la sécurité. Cet article traite d'abord des problèmes de sécurité liés à ce type de transactions, notamment ceux apparaissant au niveau de l'OS du téléphone mobile. Après un examen des techniques et approches actuelles et de l'état de l'art, il conclut sur la difficulté d'établir la confiance pourtant nécessaire à ce genre de transaction. Cet article introduit donc une nouvelle solution de sécurité indépendante des problématiques de sécurité de l'OS du mobile. Après une discussion sur divers cas d'usage et sur les aspects faisabilité, il conclut sur les avantages comparatifs de cette solution, et les travaux à venir.

Mots clés: : security, smartphone, mobile-transactions, virtualization, captcha

I. INTRODUCTION

Le marché des smartphones est entrain d'exploser. Il couvre actuellement plus de la moitié du marché des téléphones mobiles.

Les utilisateurs adoptent ce nouveau type de mobiles à cause de la grande variété de leurs usages potentiels : en plus des fonctions traditionnelles des téléphones mobiles, ils offrent les jeux, l'Internet, les applications de réseaux sociaux, l'échange de documents images ou vidéos, toutes les applications de la géo-localisation, etc.

Une nouvelle tendance est apparue depuis quelques années, avec les usages du smartphone pour des transactions de la vie quotidienne. Elle va être accélérée avec l'apparition de la technologie NFC [1] (near field communication) et le smartphone pourrait à l'avenir remplacer la carte à puce traditionnelle que nous utilisons pour par exemple payer chez un commerçant ou prendre les transports en commun, avec l'avantage d'un seul appareil pour divers types de transactions, et ceci avec des possibilités de dialogue et d'assistance de l'utilisateur beaucoup plus riches. Des exemples d'applications sont le paiement sous diverses formes de diverses prestations, le "home banking", le "ticketing", la fidélité ("loyalty"), etc, ceci dans un mode face à face (grâce au NFC) ou distant (grâce à la connectivité GSM/UMTS, ou WIFI). Etant donné les caractéristiques du marché de la téléphonie mobile et le taux de renouvellement important poussé par les approches marketing dynamiques des opérateurs, le développement du « NFC-contactless » devrait être rapide et massif, et au début 2012, tous les fabricants importants de mobiles annoncent la « NFC-isation » de leur gamme de produits ; les premières offres commerciales sont apparues fin 2011.

Cette évolution ne provient pas seulement des utilisateurs ou du marché du mobile, mais elle est aussi due aux fournisseurs de services qui voient dans l'avènement du mobile comme outil transactionnel un moyen d'améliorer la relation client, ou de simplifier ou réduire le coût de certaines opérations ; c'est par exemple le cas du renouvellement d'un abonnement aux transports en commun où l'utilisateur n'a plus à faire la queue à un guichet ou un automate.

Cependant, les besoins de sécurité relatifs à ces nouveaux usages sont importants, et difficiles à satisfaire. La section 2 est une revue de l'état de l'art actuel des techniques de sécurité employées dans les smartphones, et considère particulièrement l'OS du smartphone ainsi que son « secure element » (SE).

La section 3 est l'objet principal de cet article, et décrit la nouvelle approche sécuritaire qui ne nécessite aucune hypothèse préalable quant à la sécurité de l'OS du smartphone.

II. RÉALISATIONS ACTUELLES

A. Le modèle applicatif des plateformes mobiles

Comme les PCs, les smartphones ont un "Operating System" (OS) qui est chargé au démarrage au terme d'une procédure de démarrage ("boot") à partir d'une mémoire de masse permanente utilisant la technologie "flash. Les Smartphones sont vendus avec un certain nombre d'applications de base, et leur utilisateur a la possibilité d'en télécharger d'autres développées par diverses tierces parties, et mises à disposition sur des magasins d'applications, tels que l'Apple Store ou l'Android Market. Cet ensemble d'applications se base sur la couche « middleware », qui utilise les services de l'OS.

Les aspects sécurité concernent l'OS, mais aussi le middleware : y-a-t'il suffisamment de contrôles sur ce que fait telle application, et suffisamment d'isolation entre elles pour empêcher à des applications malhonnêtes de perturber le fonctionnement de l'OS ou d'autres applications.

Heureusement, les problématiques de sécurité sont adressées largement par les différents types de plateformes mobiles, mais elles diffèrent grandement entre elles. L'article référencé en [4] contient une synthèse sur ce sujet.

ANDROID est une vraie plateforme ouverte puisque toute application développée dans les règles sur le « toolkit android », disponible librement, peut être chargée sur un smartphone android, et rendue disponible dans un magasin d'applications officiel ou privé, au contraire d'Apple qui procède à certaines vérifications (déontologiques et sécuritaires) des applications avant de les rendre disponibles dans l'Apple-store. Ces vérifications ne semblent cependant pas constituer un filtrage très efficace de malwares : CF l'exemple [20].

Les applications du middleware (ANDROID ou iOS d'APPLE) se basent sur le langage fortement typé et contraint qu'est le langage interprété JAVA, exécuté par une machine virtuelle JAVA. Dans le cas d'ANDROID, il s'agit de la machine virtuelle Dalvik [6] qui gère, comme son équivalent dans iOS, un certain nombre de permissions et de règles, à travers une API spécifique. Les permissions concernent des opérations telles que l'établissement d'un appel téléphonique, l'envoi de SMS, l'accès au NFC, l'accès et le partage de ressources telles que le stockage dans des fichiers permanents des données de programmes, etc. Une API cryptographique est aussi disponible. L'article [5] décrit les fonctions sécuritaires du middleware du système ANDROID. Un autre exemple de middleware très utilisé est JAVA J2ME MIDP [7]. La machine virtuelle MIDP offre des fonctions équivalentes à celles citées précédemment, avec un ensemble de règles pour accorder ou pas des permissions à une application. Pour obtenir un certain niveau de permissions, l'application (Midlet) doit être signée par une autorité dont la clé publique est stockée dans le smartphone, qui est ainsi capable de vérifier cette signature. Différentes autorités peuvent exister, avec pour chacune un ensemble spécifique de permissions qu'elles peuvent accorder. Cette approche diffère d'ANDROID où une application, lors de son chargement dans le smartphone, affiche son « manifeste » qui décrit les permissions dont elle a besoin, à charge pour l'utilisateur d'accepter ou pas ce chargement, ce qui évidemment requiert de sa part un certain niveau de conscience technique! Le dispositif de signature dans ce cas est un simple moyen d'identification de l'auteur de l'application et des outils qu'il utilise.

B. La sécurité des smatphones pose déjà un problème

De nombreux exemples de défaillances de sécurité sont apparus depuis 2010. Par exemple, JailbreakMe.com est un malware bien connu qui permet dans iOS de bypasser le mécanisme de contrôle des signatures des applications téléchargées, donc de charger des applications ailleurs que sur l'Apple-Store, souvent de façon gratuite. Ce malware permet également d'inhiber le « sim-lock » du smartphone. Il est rendu possible à cause d'un manque d'isolation entre l'OS et des fichiers « pdf » contenant des scripts.

ANDROID a aussi son lot de difficultés. Un "Trojan horse" appelé « Trojan-SMS.AndroidOS.FakePlayer.a » caché dans le code d'un "player" multimédia du Google-Store, et envoyait des SMS à un serveur surtaxé, ce qui constituait un vol caractérisé du propriétaire du smartphone. Zeus est un autre exemple plus récent de ce type de malversations, basées sur la consommation non sollicitée par l'utilisateur de ressources télécoms. L'économiseur d'écran Jackeey, lui, n'attentait qu'à la "privacy" de l'utilisateur en envoyant ses informations personnelles à un serveur. Enfin le système de gestion de licences proposé dans l'API ANDROID a été cassé dés 2010 [18].

Tous ces exemples démontrent que même si les smartphones sont des produits relativement nouveaux, ils ont déjà fait l'objet de nombreuses attaques par des hackers organisés et inventifs. Certes, seules certaines de ces attaques sont basées sur des faiblesses de l'OS, beaucoup n'étant dues qu'à l'inattention ou l'imprudence de l'utilisateur. Mais étant donné les perspectives de croissance de ce marché (supérieur au marché des téléphones mobiles simples en 2011 !), étant donné l'explosion du téléchargement d'applications pour smartphone, il semble clair que la sécurité des smartphones est un problème réel et sérieux. Certaines prévisions sont, à cet égard, inquiétantes telles que celles rapportées figure 1 (origine : Kaspersky Labs) :
http://www.securelist.com/en/analysis/204792186/IT_Threat_Evolution_Q2_2011)
.

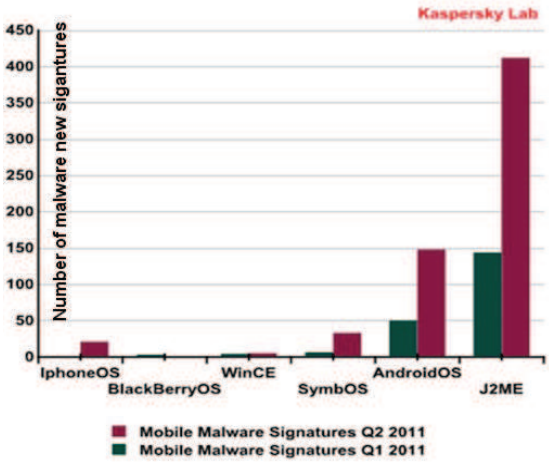

Figure 1: les tendances d'apparition de malwares pour les smartphones

Les antivirus constituent une première réponse à ce problème, mais s'ils sont efficaces, ils le sont plutôt après l'apparition et la détection de l'attaque, donc après le constat des premiers méfaits. Et comme le but des hackers est plutôt de mettre en cause la réputation des institutions, et plus rarement l'appât du gain, l'efficacité des antivirus dans le cas des applications sensibles considérées ici semble illusoire.

Une analyse sérieuse de ce que fait une application devrait être réalisée avant son apparition dans le magasin d'application De plus une application qui a été vérifiée devrait être reconnue par le smartphone qui la télécharge, accompagnée d'une description de ses caractéristiques et des ressources qu'elle peut utiliser. Ceci signifie que comme dans iOS ou MIDP, et à la différence d'ANDROID les droits d'accès à l'API ne doivent pas être seulement déclaratifs, mais subordonnés à un contrôle par une autorité de confiance reconnue, grâce à des

dispositifs de certificats et signatures garantissant l'origine et l'intégrité de ces permissions.
Cependant, même avec de telles précautions, il subsiste des risques dus à des erreurs ou des faiblesses de l'OS du smartphone, ou des possibilités de patch du code de cet OS ou/et d'applications de base lors du boot et du chargement par modification de la procédure de boot et introduction de rootkits avec des hauts niveaux de privilèges et possibilités de contrôle à distance.

Il est important de noter que les smartphones contiennent dans leur OS et dans les applications de base initialement installées (comme le navigateur par exemple) une compilation de modules et bibliothèques provenant non seulement du fabricant de ce smartphone, mais aussi de nombreuses tierces parties.
Ceci implique une difficulté extrême pour établir la confiance dans un assemblage aussi complexe et hétéroclite!
La premier exemple décrit ci-dessus correspond clairement à cette problématique !
Et à l'évidence, les hackers vont déployer de plus en plus d'efforts et d'ingéniosité pour construire des malwares ! Avec les smartphones et l'apparition d'applications sécuritairement sensibles , un magnifique terrain de jeu s'ouvre à eux !

C. Comment faire confiance aux smartphones

Cette question est difficile. Quelques éléments de réponse sont donnés ci-dessous:

- L'OS du smartphone, ses applications de base et son middleware sont d'abord conçus et développés pour fournir à l'utilisateur une interface riche (UI : user interaction) et intuitive. Ils ont des caractéristiques telles que le multitasking, des drivers pour les différentes interfaces, gestion de fichier, applications de base, et tout ceci en préservant un certain degré d'ouverture.
- Il n'est pas possible d'envisager une évaluation de type critère commun (CC) [8] avec un niveau d'assurance et d'exhaustivité suffisant car le software est beaucoup trop complexe, lourd et disparate (comparé par exemple à une carte à puce objet simple muni d'un logiciel limité et d'origine contrôlé). Un haut niveau d'assurance peut nécessiter une analyse du code source, qui bien sûr suppose l'adhésion de son éditeur.
- La confiance dépend aussi de l'adhérence entre OS et hardware, puisque beaucoup d'attaques proviennent d'un comportement anormal du hardware lorsqu'il est soumis à certaines situations, ce qui ajoute un niveau d'infaisabilité à une évaluation CC.
- De plus, la confiance en la pile logicielle du smartphone implique que son intégrité soit garantie. L'intégrité de la procédure de boot doit donc être garantie. Le TCG (Trusted Computing Group) [9] a défini une approche appelée "secure boot process" valable pour les PCs comme les mobiles. Un élément hardware inamovible et sûr appelé "mobile trusted module" (MTM) compare le hash des modules chargés lors du boot à des valeurs prédéfinies, le résultat de cette comparaison pouvant conditionner l'accés à des secrets nécessaires au bon fonctionnement du smartphone, et permettant de délivrer à une entité externe des preuves que la configuration est authentique et saine. .
- Pour des niveaux d'assurance faibles, les spécifications du smartphone et de son OS et la réputation de l'entité responsable du développement et fabrication pourraient suffire. Mais l'expérience du monde PC est plutôt négative à cet égard, et un niveau d'assurance bas ne semble pas suffisant dans le cas de certaines applications sensibles.

D. SE: secure element

Dans le monde GSM et 3G ou 4G, tous les mobiles contiennent une SIM (suscriber identity module) pouvant être considérée comme un secure element (SE). Les opérateurs de mobiles (Mobile network operators : MNO) considèrent généralement que leur SIM a les caractéristiques adéquates pour supporter les transactions mobiles sécurisées. Une des raisons concerne le marketing, puisque l'hébergement d'applications comme le paiement, le ticketing, .. peut leur fournir de nouveaux revenus et réduire le « tchurn » , c'est-à-dire l'habitude des clients de changer d'opérateur pour bénéficier d'offres plus attrayantes.
L'autre raison est technque, car la SIM (ou UICC/USIM pour la 3G) est une carte à puce complexe conforme aux standards ETSI ou 3GPP et basée aussi sur les normes Java Cards [11] et Global Platform [12].
Elles ont donc les caractéristiques nécessaires pour supporter le « multi-application », c'est-à-dire l'isolation entre applications émanant de différents émetteurs d'applications, la possibilité de charger et installer ces applications, y compris leurs clés, en cours de vie de la carte, la possibilité de supprimer des applications, et tout ceci dans de bonnes conditions de sécurité. La confiance en ces caractéristiques peut être établie par des évaluations CC.
Les évaluations CC sont une méthodologie bien connue et pratiquée dans le monde des cartes à puces.
En France, l'AFSCM [19] facilite le développement du sans contact mobiles en établissant des règles de fonctionnement et de sécurité et des spécifications concernant différents projets d'applications mobiles sans contact, sous le nom grand public « CITYZI ». Les besoins sécuritaires établis avec les banques Françaises stipulent une niveau d'assurance EAL4+ basé sur un « profil de protection » (listant les caractéristiques sécuritaires demandées) détaillées dans les références [13] et [14].
D'autres formes de SE peuvent exister : certains fabricants logent le SE à l'intérieur du chip NFC qui assure l'interface sans contact avec un autre terminal ou carte. Une autre solution existe à travers les cartes « secure SD ».
Chacune de ces alternatives se développera (ou pas) en fonction des forces et tendances du marché, et se conformera aux standards mentionnés ci-dessus. Ainsi, malgré des différences dans le hardware, toutes les conditions sont réunies pour avoir une approche commune de l'architecture sécuritaire

logique du smartphone, indépendante de l'implémentation hardware du SE.

E. Architecture de sécurité

Conformément au propos précédent, on suppose donc ici que le smartphone contient un SE protégeant les « biens » sensibles des fournisseurs de services émetteurs d'applications, comme des clés cryptographiques ou des fonctions sensibles telles que celles prenant la décision d'accepter un paiement off-line au vu des paiements déjà effectués.
Ces fonctions sont extrèmement sensibles, car une attaque réussie pourrait par exemple conduire à la création de fausse monnaie. Dans certains cas, des fonctions sensibles sont réalisées dans le smartphone, car demandeuses de trop de puissance de calcul ou de capacités de stockage : c'est par exemple le cas des DRMs (digital right management) où le contrôle de droit et le calcul des clés de déchiffrement du contenu peuvent être réalisés dans le SE, mais le déchiffrement du contenu est plutôt réalisé sur la plateforme.
Comme la confiance à un haut niveau d'assurance est possible avec le SE, nous ne considérons pas ici les attaques sur ce SE.

F. Problèmes de sécurité restants

Nous considérons donc uniquement les attaques sur la partie des applications sensibles implémentées dans le smartphone.

Une attaque envisageable est qu'un malware déclenche par exemple un paiement sans l'accord de l'utilisateur, à la manière des virus décrits plus haut. Un autre exemple d'attaque est de tromper l'utilisateur en affichant des informations erronées sur l'écran. Ceci correspond par exemple au problème nommé "what you see is what you pay".
Le premier exemple nécessiterait la connaissance du PIN utilisateur, qu'un virus pourrait capter à la manière d'un « key-logger » ou plus simplement par une attaque de type « phishing », où l'utilisateur fournit son PIN à une application malveillante qui n'est pas la vraie application de paiement.
Notons que dans le cas des sytèmes traditionnels de paiement, ces attaques sont impossibles car le terminal commerçant et son « PINPAD » sont considérés comme sûrs.
Certes Global Platform est en train de définir un système de contrôle d'accès applications smartphones-applications SE (applets) mais l'authentification par le SE ne pourra se baser que sur une procédure simple de vérification de signature de l'application smartphone, qui à partir du moment où l'on considère que la sécurité de l'OS du smartphone est faible sera elle-même faible, à moins de se conformer au modèle de sécurité du TCG.
La présence du SE ne garantit donc pas contre certaines attaques et l'utilisateur du smartphone pourrait être trompé ou volé par certains types de malwares, ce qui aurait évidemment des conséquences facheuses en terme de réputation des systèmes de paiement et de leur sécurité, et donc détruirait la nécessaire confiance en ce type de moyen de paiement.

G. Comment améliorer la sécurité:

Comme mentionné plus haut, avoir un OS+ middleware de confiance sur un smartphone semble illusoire.
Cependant de nouvelles approches apparaissent, généralement basées sur le concept de virtualisation. La référence [12] synthétise cette nouvelle approche.
Il est possible par exemple de combiner un OS classique et « riche » (sur le plan des UI et des nombreuses applications gérées) avec un OS minimal stocké dans une partie sûre de la plateforme smartphone. Les applications sensibles seraient chargées et installées après contrôle de leurs signatures dans cette partie sûre. L'OS minimal gèrerait de façon sûre les entrées sorties en priorité sur l'OS riche, et il serait de taille et complexité limitée, ce qui permettrait un genre d'évaluation sécuritaire au sens CC. Le groupe de standardisation Global Platform [12] travaille sur une approche de ce type, appelée TEE (trusted execution environment). Cependant, la complexité de ce type de solution semble grande, puisque un travail de standardisation important doit être réalisé pour définir des requis fonctionnels et sécuritaires:
-l'interface entre l'OS riche et l'OS minimal pour obtenir une bonne indépendance applications/platefortmes,
-détails de l'interface entre l'OS minimal et le hardware pour obtenir une bonne sécurité des UI,
-moyens de garantir l'intégrité de l'OS minimal et preuve de cette intégrité au monde extérieur,
Tout ceci dans un contexte complexe où des groupes industriels géants peuvent poursuivre leurs propres stratégies.

En conclusion de la section II, il n'y a pas aujourd'hui de solution réellement satisfaisante sur le plan sécurité, et l'obtention d'une solution convaincante sur le plan confiance nécessite un accord entre la multitude des acteurs du marché des smartphones, tels que les fabricants, les MNOs, les fournisseurs de service, ce qui risque prendre du temps, et une fois la solution disponible, impliquer des coûts et délais importants puisque nécessitant un minimum d'évaluation sécuritaire d'un dispositif extrêmement complexe.

III. UNE SOLUTION SIMPLE

A. Captcha

Les CAPTHA (Completely Automated Public Turing test to tell Computers and Humans Apart) sont un moyen bien connu et largement utilisé pour protéger contre des attaques visant à remplacer un utilisateur humain par un automate pour accéder à un serveur sur le Net. Les Captchas se basent donc sur un test de reconnaissance de certains signes ou formes par un humain qu'un automate ou un programme d'ordinateur ne peuvent passer dans l'état actuel des technologies.
Par exemple un utilisateur humain reconnait facilement un texte écrit en utilisant des caractères distordus, parasité par différents dessins annexes, ce dont est incapable un logiciel de reconnaissance de caractère. Une publication synthétique sur le sujet est donnée en [15].

B. *Architecture de confiance*

On considère le cas d'une transaction entre un utilisateur, qui dispose d'un smartphone, et un fournisseur de service qui dispose de son propre terminal ou serveur communiquant avec le smartphone via NFC ou Internet. Pour l'utilisateur, le seul élément de confiance est le SE dans le smartphone, et la figure 2 montre la manière d'établir la confiance nécessaire à la transaction :

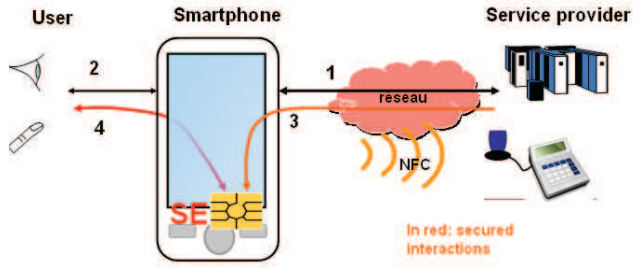

Figure 2: modèle de sécurité

- La confiance dans les messages (1 et 2) entre utilisateur et le fournisseur de service (SP) ne va pas de soi, du fait des diverses attaques possibles dans le smartphone, décrites dans la section II.
- La confiance en les messages (3) entre SP et SE est assurée par des protocoles de communication sécurisée tels que décrits dans GP, la sécurité étant de plus «de bout en bout » entre SP et SE, donc indépendante du smartphone.
- La confiance entre le SE et l'utilisateur (4) est garantie par des Captchas fabriqués dans le SE.
- Donc, la confiance des échanges entre SP et l'utilisateur devient possible si les contenus des messages 1 et 2 sont confirmés par le contenu des messages 4.

Notons que dans cette approche, les messages 4 sont bi-directionnels: ils se basent sur le principe du captcha mais permettent également la protection d'informations saisies par l'utilisateur. Nous appelons ZIS (Zone d'Interaction Sécurisée) cette extension de l'approche Captcha. Nous décrivons dans la suite deux cas d'usage qui même s'ils n'ont rien à voir, font usage de ce concept de ZIS tel décrit sur la figure 3[1].

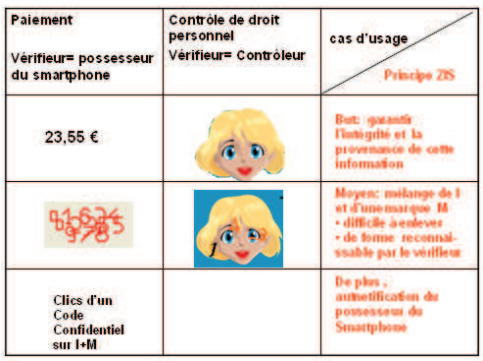

Figure 3: l'approche ZIS dans deux cas d'usage

[1] Brevets # 10 02516 et # 11 01369

C. *Cas d'usage paiement.*

Ce cas peut, comme le montre la figure 4, se formaliser en une proposition de contrat envoyée par le SP (FNAC Caen dans l'exemple) à l'utilisateur (un BRDVD valant 23,55€). Ces données, accompagnées par les identifiants du SP, la date, des aléas, etc, sont envoyées par le terminal du SP au smartphone et son SE, via NFC, lors d'un premier « tap » entre smartphone et terminal.

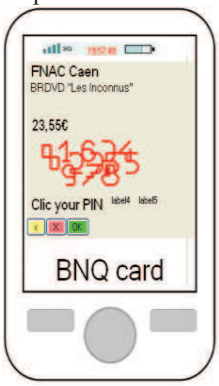

Figure 4 : cas d'usage paiement

L'application Captcha, dans le SE, fabrique donc une ZIS sous forme graphique. Sur la figure 4, la ZIS est le dessin apparaissant en rouge. Elle contient 3 lignes de chiffres, la deuxième ligne répétant le montant (23,55) et les autres lignes contenant des chiffres complémentaires permettant la saisie du PIN (les 10 chiffres doivent donc apparaître) et se superposant à la ligne du milieu pour rendre difficile la reconnaissance de caractères par un malware. L'utilisateur entre son PIN en cliquant sur 4 chiffres de la ZIS correspondant à ce PIN. Si le PIN est correct, un second tap déclenche l'application signature dans le SE, qui constitue donc un engagement de l'utilisateur à payer ce montant à ce prestataire grâce à une signature cryptographique des données de la transaction reçues par l'application captcha lors du premier tap.

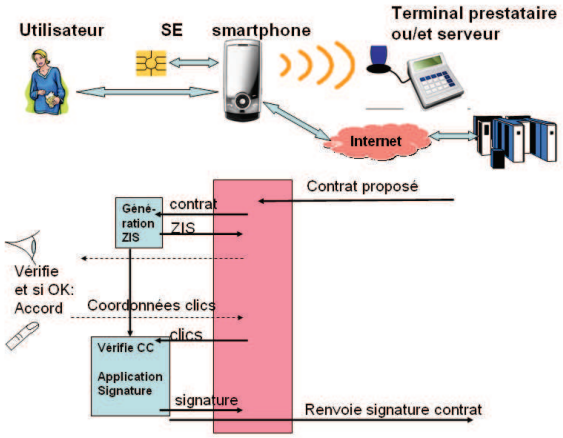

Figure 5: cinématique des différents messages

La figure 5 représente la cinématique des échanges entre les différents éléments du système

D. *Robustesse de la méthode:*

Il est bien connu que dans certains cas, les captcha peuvent être cassés.: Voir par exemple l'initiative deCAPTCHA [17] de l'université de Stanford, qui a réalisé une analyse exhaustive des réalisations sur ce sujet. Mais la fragilité d'une technique Captcha donnée dépend de nombreux aspects techniques et usage de cette implémentation. Dans notre cas, nous pensons que les caractéristiques énoncées ci-dessous apportent un niveau de sécurité suffisant :

1. les chiffres peuvent être dessinés par l'utilisateur du smartphone, avec un petit programme d'édition. Leur forme est donc totalement dépendante de l'utilisateur, de son imagination et fantaisie. Ceci contre des attaques de type « phishing » où un malware établirait sa propre ZIS, afin de voler le code confidentiel.
2. La superposition entre les 3 lignes rend difficile pour un malware la compréhension du Captcha, et l'apprentissage des caractères de ce smartphone.
3. De plus, des déformations aléatoires sont appliqués sur les chiffres du Captcha: les tailles et inclinaison varient entre deux ZIS et à l'intérieur même d'une ZIS.
4. La modification de la ZIS par un malware nécessiterait l'effacement, la modification et l'addition de certains chiffres dans la ZIS. Ceci provoquerait des artefacts et anomalies qui seraient probablement détectées par l'utilisateur, et qui de plus modifieraient l'entrée du PIN..
5. La protection en confidentialité du PIN provient de ce que les coordonnées des clics sont relatives à la ZIS, qui est incompréhensible à un malware dans le smartphone.
6. De plus, un malware attaquant la ZIS est limité par le temps de réponse exigé par le SE, qui gère un time-out de quelques secondes

E. *Cas d'usage contrôle de droit personnel*

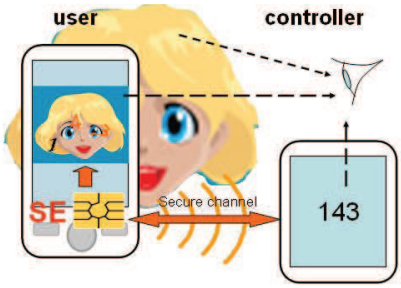

Figure 6: smartphone comme moyen d'identification

Souvent, un utilisateur doit prouver qu'il a un abonnement à tel ou tel service, ou qu'il a tel privilège (cas des applications de fidélité).
La méthode habituelle est de fournir à l'utilisateur une carte papier ou plastique avec sa photo, prouvant la possession de ce droit.
Le smartphone pourrait remplacer utilement toutes ces cartes, mais alors se pose le problème de la sécurité, qui peut être aigu dans le cas d'un droit personnel de valeur importante, comme par exemple un abonnement pour le transport public. En effet rien n'est plus facile que d'afficher une photo quelconque sur l'écran d'un smartphone !
La figure 6 est relative à ce type de cas d'usage où le smartphone de l'utilisateur est utilisé pour le contrôle d'un droit par un contrôleur humain, qui vérifie la photo relative à ce droit. Le contrôleur a un terminal qui communique en NFC avec le smartphone. La solution qui consisterait à envoyer la photo avec une preuve d'authenticité (signature du hash de la photo) du smartphone vers le terminal du contrôleur est simple mais insatisfaisante sur le plan « privacy ».
Dans notre solution, la photo affichée sur le smartphone contient une marque (nombre "143" sur la 6). La photo est stockée dans le SE, et le processus de marquage est réalisé dans ce SE. Pour empêcher des attaques de type rejeu, la valeur de la marque est générée aléatoirement par le SE et/ou le terminal contrôleur, avec des échanges SE-terminal confidentiels. La valeur de la marque est affichée sur le terminal contrôleur et elle correspond au challenge d'un protocole d'authentification. Le contrôleur peut donc vérifier que la marque sur la photo affichée sur le smartphone correspond bien à la marque affichée sur son terminal, ce qui prouve l'authenticité de la photo puisque du côté du smartphone, seul le SE connaissait la valeur de cette marque.
On utilise également dans ce cas d'usage le concept de ZIS, apportant une sécurité de bout en bout entre le SE du smartphone et l'utilisateur, qui au sens du concept ZIS est ici le contrôleur.

F. *Robustesse du marquage*

Un malware dans le smart phone qui essaierait de remplacer la photo I par une autre, I', aurait d'abord à réaliser une analyse d'image de l'image marquée I+M, de façon à en extraire la marque M, qu'il collerait ensuite sur l'image I' pour obtenir ainsi I'+M qui serait authentifiée et acceptée par le contrôleur, permettant ainsi à I' d'utiliser les droits de I. Le malware aurait à utiliser des technqiques de traitement d'image classiques (edge detection , pattern matching). Tout ceci devrait être réalisé dans les quelques secondes laissées par le terminal contrôleur pour l'affichage de la photo et l'accord ou le refus du contrôleur. Pour améliorer la sécurité, nous utilisons des techniques de « floutage » et de transparence dans le processus de marquage, de façon à rendre plus difficile l'isolation de M dans I+M

1) *Implémentation*

La bonne approche sécuritaire est que la génération du captcha soit faite dans l'élément de confiance qu'est le SE. Notons que d'autres modes de réalisation sont possibles si le smartphone utilise des concepts tels que virtualisation ou autre « trusted zone » qui permettraient même de lever la difficulté que représente la faible puissance de calcul du SE pour des opérations de type graphique.

Néanmoins, les deux cas d'usage ont été développés avec les deux applications du SE (génération du captcha du cas 1 et marquage photo du cas 2) sous forme d'applet Java Card, et nous obtenons un niveau de performances satisfaisant: 1 seconde pour la cas 1 et 2 à 8 secondes dans le cas 2, selon certains choix d'implémentation.

Pour le cas 2, une des difficultés résidait dans le processus de marquage, qui peut être un processus de traitement d'image lourd et complexe et qui doit être réalisé dans le SE. Si ce processus était réalisé au niveau pixel dans le SE, il nécessiterait beaucoup trop de temps et d'espace puisque une photo noir et blanc 400x400 en 256 niveaux de gris nécessite 160kilo octets de RAM, et qu'un SE ordinaire est limité à quelques dizaines de kilo octets de RAM.

Le processus de marquage est donc réalisé directement sur l'image compressée (12 kilo octets) où nous utilisons une forme spéciale de compression de type JPEG pour faciliter le marquage qui, de fait, devient un simple processus de patch.

En conclusion, le choix de l'implémentation Java Card est important car il facilite le déploiement de cette solution sur divers SE sans se soucier de leur constitution hardware et software de bas niveau, et de plus il permet d'utiliser les infrastructures de téléchargement d'applets « over the air » et du software associé dans le smartphone.

IV. CONCLUSION & FUTURS TRAVAUX

Cet article a décrit une solution simple essentiellement basée sur l'élément de confiance qu'est le SE, et qui résout tous les problèmes complexes de sécurité décrits section II, qui semblent difficiles à traiter dans le contexte smartphones.

Cette solution est totalement indépendante de l'OS et middleware des smartphones et de leur sécurité. Elle est également indépendante des protocoles entre smartphone ou SE et serveurs.

Ainsi, elle peut être implémentée sur une base concurrentielle, puisque ne nécessitant pas de conformité à un standard.

Les faibles capacités de calcul du SE n'empêchent pas d'atteindre des niveaux de performances suffisants.

Cette solution ne peut évidemment prétendre à une quelconque universalité. Cependant elle peut intéresser la catégorie très large des applications de type paiement et contrôle d'accès. Il est clair qu'elle peut aussi intéresser le monde du PC où le SE peut prendre différentes formes.

Divers démonstrateurs ont été réalisés, sur cartes Java, sur PC et sur smartphone NFC ANDROID afin d'en valider les performances et l'ergonomie. Les travaux futurs porteront sur la réalisation de démonstrateurs plus réalistes et aussi sur l'évaluation de l'acceptabilité de ces approches par l'utilisateur grand public, grâce à des tests avec un panel d'utilisateurs significatif. En effet, le supplément de sécurité obtenu par le procédé décrit dans la section III a bien évidemment une contrepartie en terme de complication pour l'utilisateur qui doit supporter des interactions un peu moins simples avec son smartphone !

References

[1] NFC: see NFC Forum: http://www.nfc-forum.org/home

[2] MOBEY Forum: http://www.mobeyforum.org/

[3] http://www.nearfieldcommunicationsworld.com/2010/06/17/33966/all-new-nokia-smartphones-to-come-with-nfc-from-2011/

[4] Alan Goode. Managing mobile security: How are we doing? Network Security, 2010(2):12–15, 2010.

[5] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding android security. IEEE Security & Privacy, 7(1):50–57, 2009.

[6] Dalvik Virtual Machine : http://developer.android.com/intl/fr/guide/basics/what-is-android.html

[7] Mobile Information Device Profile (MIDP); JSR 118; http://www.oracle.com/technetwork/java/index-jsp-138820.html

[8] Common criteria : http://www.commoncriteriaportal.org/cc/

[9] AFOM: http://www.afom.fr

[10] TCG: https://www.trustedcomputinggroup.org/home; « Trusted computing platforms, trusting computing group"

[11] JAVA CARD : see the forum http://www.javacardforum.org/

[12] http://www.vmware.com/fr/

[13] GLOBAL PLATFORM: http://www.globalplatform.org/

[14] (U)SIM protection profile: http://www.ssi.gouv.fr/site_rubrique92.html

[15] Luis von Ahn, Ben Maurer, Colin McMillen, David Abraham and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. In *Science.*

[16] Kumar Chellapilla and Patrice Y. Simard. Using Machine Learning to Break Human Interaction Proofs (HIPs). In *NIPS.*

[17] deCAPTCHA project: http://decaptcha.net/

[18] DRM Android: http://www.androidpolice.com/2010/08/23/exclusive-report-googles-android-market-license-verification-easily-circumvented-will-not-stop-pirates/

[19] AFSCM: http://www.afscm.org/

[20] http://articles.businessinsider.com/2011-11-09/tech/30376688_1_ios-security-flaw-cnn

.

Abbreviations:

- SP service provider
- MNO mobile network operator
- NFC near field communication
- UI user interaction
- SE secure element
- SIM subscriber identity module
- GP global platform standard
- SMS: short message service
- PIN: personnal identification number
- identification number