
CryptoPage-1

Vers la fin du piratage informatique ?

Ronan Keryell

Laboratoire d'Informatique des Télécommunications
École Nationale Supérieure des Télécommunications de Bretagne

EUROSEC'2001

15 mars 2001

Introduction

1

- Développement de média de forte capacité bon marché :
CD-ROM, DVD,...
- Permettent diffusion de contenus numériques : logiciels,
artistiques (films, musique)
- Faible coût de production échappant aux lois économiques
classiques :
 - ▶ Recopier simplement des suites de « 0 » et de « 1 »
 - ▶ Coût concentré dans la confection de l'original plutôt que
dans les copies
- Démocratisation d'Internet et des réseaux rapides : possibilité de
télécharger des contenus virtuels
- Développement de média en version inscriptible et faible prix du
disque dur

~> ∃ Débordements du cadre de la copie privée de sauvegarde...



- Besoin de systèmes résistants
 - ▶ Banques, distributeurs de billets
 - ▶ Commerce électronique
 - ▶ Systèmes défense nationale
 - ▶ Réseaux sécurisés
- Attaques possibles
 - ▶ Interception des communications ou des bus
 - ▶ Injections de fausses données (bus, mémoires, ports, réseaux,...)
 - ▶ *Glitches*
 - ▶ Rétro-ingénierie (décapage chimique, micro-sonde ionique,...)
 - ▶ Accélérateur de particule
 - ▶ ...



- Principe de base
- L'existant
- CryptoPage-1
 - ▶ Matériel
 - ▶ Logiciel



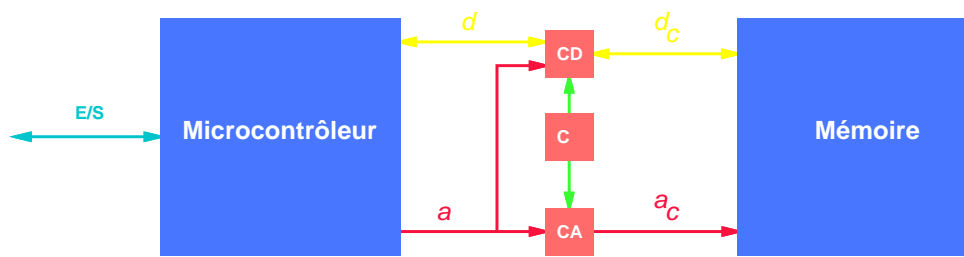
- Spécialisation du processeur : devient unique au monde
 - ▶ Utilisation de cryptographie à clé publique/clé secrète
 - ▶ Producteur du logiciel crypte avec la clé publique du processeur
 - ▶ Processeur exécute le programme en déchiffrant avec sa clé secrète
 - ▶ Impossible de déchiffrer avec la clé publique
- Communications secrètes
 - ▶ Algorithme de routage secret
 - ▶ On ne sait plus ce qui passe sur la ligne : bruit ou données?
- Contenu numérique artistique
 - ▶ Même si crypté, piratage possible lors de la matérialisation en son ou image
 - ▶ Possibilité de crypter le contenu multimédia associé (DVD,...)



- Utilise cryptographie forte
- Sécurité garantie par des mécanismes physiques de protection (détection d'ouverture,...)
- Ordinateur embarqué complet mais... petit
- Cadre trop restreint pour un ordinateur standard : besoin de
 - ▶ Vitesse > 1 GFLOPS
 - ▶ Grosse mémoire
 - ▶ Bus rapide(s)
 - ▶ Périphériques rapides (disques, écran)
 - ▶ Système d'exploitation standard





- Chiffrement des bus du processeur
- Exemple du DS5002FP (8051 sécurisé pour terminaux cartes bancaires) :



- Pour éviter les attaques à texte connu (zones de 0...)



$$d_c = C_D(c_s, a, d)$$



- ▶  Le circuit a été « cassé »
 C_D est bijective : toute instruction lue en mémoire va être exécutée. Jeu d'instruction sur 8 bits : 256 essais d'injection à faire après le RESET \leadsto construction de proche en proche d'un programme sortant sur un port la mémoire déchiffrée !
- Dans les circuits proposés par Robert M. Best toute instruction invalide provoque la destruction du processeur.  bugs, virus,...

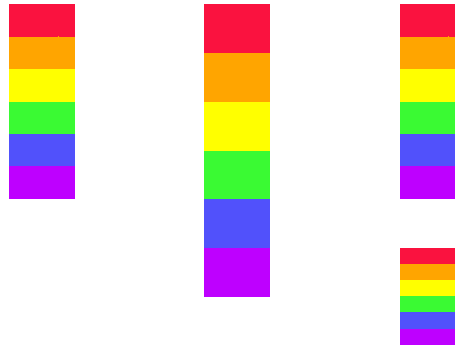



- Rajouter une signature électronique à des « Pages » ou lignes de cache de données
- Au lieu de chiffrer d on chiffre $d \parallel H(d)$
- Au déchiffrement on a $d'' \parallel h''$ et il suffit de vérifier que $H(d'') = h''$
- Si H fournit des valeurs sur b_s bits la probabilité d'être trompé est de 2^{-b_s} :
pour $b_s = 128$, cela fait une chance sur $3,4.10^{38}$, raisonnable dans l'état actuel de la technologie

- Rajout de b_s bits par page de b bits...
-  Que faire de ces b_s bits supplémentaires de manière transparente ?
- Corollaire : modifier l'adressage mémoire dans le traducteur *adresse virtuelle*
  *adresse physique*
- 2 solutions étudiées
- Solution dilatée plus simple pour les programmes :
connexité est préservée (sauvegarde sur disque,...).

Une adresse a se retrouve en :

$$\left[\left\lfloor \frac{a}{b} \right\rfloor \frac{b + b_s}{b}, \left\lfloor \frac{a}{b} + 1 \right\rfloor \frac{b + b_s}{b} - 1 \right]$$



- Pas de chiffrement au niveau adresse (au moins au niveau des pages)
- Bénéficie du support système de la mémoire virtuelle
-  Faire attention aux programmes gérant simultanément des programmes/données chiffrés et en clair : éviter les conflits d'adresse \leadsto géré par `malloc()`
- Si chaînage à la CBC : encore des bits supplémentaires





- Cryptographie
 - ▶ Algorithme à clé publique utilisé pour décoder le descripteur
Peu critique et possibilité de faire des caches de descripteurs décryptés
 - ▶ Algorithme symétrique plus critique car fait le chiffrement entre le cache et la mémoire
CS Cipher par exemple : 8 Go/s à 1 GHz. Si plus : parallélisable, pipelinable,...
- Pour mieux résister
 - ▶ Faire suivre tout chiffreur par un déchiffreur et réciproquement pour vérifier l'intégrité physique
 - ▶ Faire voter plusieurs chiffreurs/déchiffreurs pour détecter *glitches*, variations de fréquence d'horloge, température, rayonnements ionisants,...



- ▶ Cible : processeurs de 10^7 ou 10^8 transistors \leadsto coût faible de la redondance

Destruction du processeur si incohérence

- Tests
 - ▶ Suppression du JTAG en mode crypté
 - ▶ Possibilité de faire un test en mode crypté en usine puis grillage de *plusieurs* fusibles
- Étudier le graphe de dépendance afin de prouver qu'une donnée chiffrée ne peut pas être dirigée vers une donnée non chiffrée




- Nouvelles instructions (version RISC)
 - ▶ RTIEC r : *ReTurn from Interrupt on an Enciphered Context*
 - ▶ LDNC r_d, r_a, r_{ac} : *LoaD Non-Ciphered*
 - ▶ STNC r_d, r_a, r_{ac} : *STore Non-Ciphered*
- Exemple dans Unix (sources disponibles, robuste et souvent libre)
 - ▶ Un processus peut être $[root, user] \times [clair, chiffré]$
 - ▶ Même $root$ (piratable,...) ne peut pas espionner un processus crypté
 - ▶ Pour partager de l'information chiffrée entre processus :
partage d'une clé commune
 - ▶ Déverminage d'un processus crypté : le dévermineur doit être crypté avec la même clé

- Un processus crypté doit pouvoir utiliser les services du noyau non crypté
- Déchiffrement *par le processus crypté* des arguments avant le trap
 - ▶ Pas d'édition de lien dynamique possible directement
 - ▶ Compatibilité système ascendante à gérer au niveau trap
- Appels systèmes exportent/importent des données chiffrées
 - ▶ Pour garder la sémantique Unix, les tailles des données sont celles des données cryptées
- Bibliothèques standards : existent en version cryptées et en clair
 - ▶ `malloc()` et `calloc()` alloue la place supplémentaire pour la signature et aligne tout sur des blocs entiers

- Cartes à puces virtuelles
- Protection des serveurs WWW,...
- *Dongles* virtuels
- Routeurs à logiciel crypté
- Machines virtuelles
- Agents secrets (code mobile, Applets,...)
- Grilles de calcul cryptées
- Antivols électroniques pour ordinateur



- Avoir une clause au contrat de vente permettant de transférer un programme vers un autre ordinateur (panne, vol,...)
-  On ne peut même plus savoir ce qui tourne sur son ordinateur
 - ▶ Bugs irréparables
 - ▶ Comportements cachés
- Mais la boîte de Pandore existe déjà
 - ▶ Mais actuellement c'est déjà difficile (OS propriétaires,...), même avec les sources (empilements logiciels gloutons,...)
 - ▶ Il y a déjà des closes interdisant la rétro-ingénierie sur des contrats de logiciels et des comportements cachés
- Rien que l'identifiant unique du Pentium III a provoqué des réactions...
- On n'a pas le droit de regarder ses DVD avec des logiciels libres



- Sujet « chaud » avec les actions judiciaires entre éditeurs/artistes et sites pirates
- Doit pouvoir empêcher le piratage par une entité non-étatique
- Coût matériel faible
- Acceptation sociale ?
- Pas de solution satisfaisante pour les contenus artistiques numériques (qui est prêt à se faire greffer des implants neuronaux ?)
- Version suivante doit empêcher les attaques par replay de vieilles données



List of Slides

- 1 Introduction

Introduction

- 2 Sécurité des systèmes
- 3 Plan
- 4 Principe

Principe

- 5 Carte à puce

Existant

- 6 Crypto-processeurs
- 8 Signature électronique

Matériel

- 9 Adressage mémoire
- 11 La grande image
- 12 Mise en œuvre
- 14 Mise en œuvre logicielle

Logiciel

- 15 Appels systèmes
- 16 Quelques applications

Conclusion

- 17 Éthique
- 18 Conclusion
- 19 Table des matières

