Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

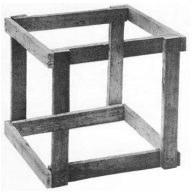
PREMIER MINISTRE

Secrétariat général de la défense et de la sécurité nationale

Agence nationale de la sécurité des systèmes d'information

Documentation CLIP 1307

TPM



ANSSI, 51 boulevard de la Tour Maubourg, 75700 Paris 07 SP.

HISTORIQUE

Révision	Date	Auteur	Commentaire
0.0	26/01/2015	Thomas Letan	Ceci n'est encore qu'un draft

Table des matières

Résumé

1 Besoins en intégrité

1.1 Préoccupations

CLIP est un système d'exploitation bi-niveaux développé par l'ANSSI depuis 200X. Basé sur la distribution Linux Gentoo, il est dit « durci », car embarquant un certain nombre de mesures visant au renforcement de sa sécurité. Ces mesures sont constituées de *patches* (externes ou internes) sur des solutions existantes et de solutions développées en interne et offrent plusieurs garanties, comme une isolation forte entre le niveau haut et le niveau bas, une protection en intégrité du code exécuté, une protection en confidentialité des données, etc.

Ces protections sont efficaces au cours de la vie du système, mais leur bon fonctionnement suppose implicitement qu'elles ont été correctement mises en œuvre au démarrage. Il s'agit d'une hypothèse forte : en effet, la partition de démarrage d'un système CLIP n'est pas protégée en intégrité. Une attaquant ayant un accès physique à un poste CLIP peut facilement forcer ce poste à démarrer sur une clef USB et monter la partition de demarrage afin de modifier le noyau CLIP dans le but de rendre inopérants ses mécanismes de sécurité. Après une telle manipulation, un poste CLIP est compromis, sans que son utilisateur légitime n'en ait conscience. Au prochain démarrage de sa machine, rien ne l'empêchera de taper son mot de passe pour se connecter à son compte utilisateur, donnant à un attaquant les informations qui lui manquait pour avoir un accès total sur la machine considérée. Un tel scénario d'attaque impliquant un accès physique à la cible est souvent appelé scénario *Evil Maid*[evil09] dans la litérature, afin de l'illustrer plus concrétement. En laissant son poste CLIP dans sa chambre d'hôtel, un agent de l'ANSSI donne un accès physique à sa machine ce dernier à quiconque possède un moyen d'accès à cette dernière.

C'est parce que CLIP dans sa version courante (4.4.1) n'est absolument pas protégé contre ce type de scénario d'attaque qu'un poste CLIP perdu puis retrouvé est considéré par défaut comme compromis et ne doit plus être utilisé. Le but du présent document est de présenter comment il est possible de détecter, au démarrage, des corruptions du code des logiciels impliqués dans le démarrage du système. Pareilles détections rendraient inefficaces le scénario précédemment décrit.

1.2 Informatique de confiance

La propriété de sécurité recherchée pour prévenir un scénario d'attaque de type *Evil Maid* est l'intégrité du noyau CLIP exécuté sur une machine. Il existe deux façons de s'attaquer à cette intégrité :

1. En modifiant directement le fichier `bzImage` stocké sur la partition d'amorçage ;
2. En attaquant un composant logiciel intervenant plus tôt dans le démarrage du poste (BIOS [1], chargeur d'amorçage, etc.) afin que ce dernier modifie dynamiquement le noyau à son chargement.

Aussi, il convient de s'assurer de l'intégrité du noyau CLIP, mais aussi de tous les composants logiciels intervenant dans la phase de démarrage.

L'informatique de confiance est un paradigme de sécurité informatique visant à s'assurer que le code chargé en mémoire (et donc exécuté par la suite) par une machine est un code « de confiance », à mettre en opposition avec un code arbitraire et potentiellement malveillant. La notion de confiance est ici à la discrétion de l'utilisateur. Il peut décider de ne considérer comme de confiance que du code dont il est lui-même l'auteur. Plus vraisemblablement, il peut n'accorder sa confiance qu'à un nombre restreint de partenaires. Par exemple et dans le cas de l'ANSSI et des postes CLIP, l'agence n'étant pas éditrice de BIOS, il est raisonnable de considérer « de confiance » le BIOS fournit avec les plateformes supportées.

Il est important de souligner qu'un code de confiance n'est absolument pas pas un code exempt de bogues ou de vulnérabilités. L'informatique de confiance n'est pas directement liée à, par exemple, la vérification

1. *Basic Input/Ouput System*

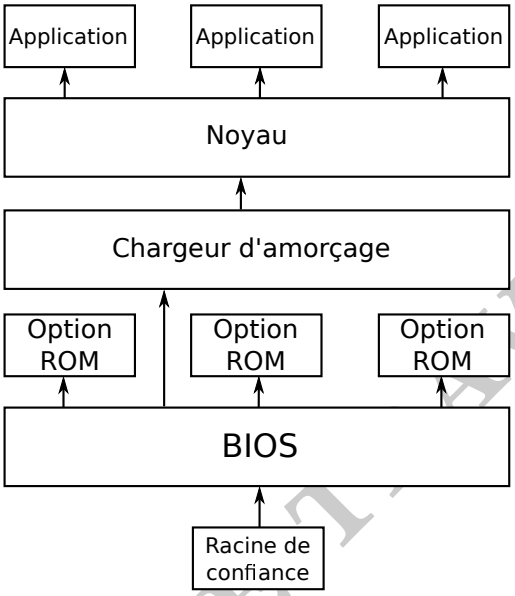


FIGURE 1 – Vue schématisée de la chaîne de confiance

formelle [2].

Tout l'enjeu de l'informatique de confiance est l'établissement d'une chaîne de confiance allant des premiers instants du démarrage de la machine à son exctinction. Tout code doit être, après son chargement et avant son exécution, reconnu comme « de confiance ». S'il ne l'est pas, alors la chaîne de confiance est brisée et le système se retrouve dans un état compromis. Il est important de souligner qu'il s'agit d'un mécanisme intrusif : les chainons de cette chaîne de confiance que l'on veut construire doivent être modifiés en ce sens. Vérifiés par l'élément qui les précèdent, ils ont la charge de vérifier celui qui les suit. Cela pose la question du traitement du tout premier code exécuté lors du démarrage, ce dernier ne bénéficiant *a priori* d'aucune vérification.

Traditionnellement, ce premier code exécuté est appelé la racine de confiance (*Root of Trust*) et jouit d'une protection renforcée. Il est inscrit dans une mémoire particulière isolée du reste de la machine et ses procédures de mise à jour sont hautement critique. Plusieurs failles de sécurité ont malheureusement été découvertes ces dernières années ; elles montrent clairement une partie des limites de l'informatique de confiance. Plusieurs méthodes ont été proposées pour renforcer la racine de confiance, par exemple par le MITRE ou HP.

La Figure 1 montre de manière schématique à quoi ressemble une chaîne de confiance.

1.3 *Secure Boot*

Le *Secure Boot* est un mécanisme de protection du demarrage de l'ordinateur présenté dans les spécifications de l'*Unified Extensible Firmware Interface* (UEFI). Il a été grandement popularisé par son usage dans Windows 8 et il tend à être supporté par la plupart des implémentations UEFI.

Le *Secure Boot* propose de différencier un code de confiance d'un code arbitraire grâce à l'usage de signatures cryptographiques. Plus qu'au code en lui-même, la confiance est ici accordée à un tiers. Le *Secure Boot* impose que chaque composant logicielle intervenant dans le démarrage soit cryptographiquement signé par son éditeur. Les éditeurs sont identifiés par un certificat liant son identité avec la clef public permettant de vérifier la-dite signature. C'est en ajoutant ou non le certificat d'un éditeur de logiciel dans les configurations

2. La vérification formelle d'un produit peut certes intervenir dans la décision d'accorder ou non sa confiance au dit produit, mais cela reste à la discrétion de l'utilisateur… Qui devra dans un premier temps décider dans quelle mesure il accorde sa confiance à la méthode de vérification formelle utilisée.

du *Secure Boot* de sa machine qu'un utilisateur lui accorde ou non sa confiance. Tout code signé par cet éditeur sera donc considéré comme de confiance.

Au démarrage, le *firmware* UEFI vérifie les signatures de tous les exécutables qu'il doit charger et exécuter. De cette manière, il y a établissement d'une chaîne de confiance. Si la vérification échoue [3], alors la chaîne est brisée et le code ne peut pas être déclaré de confiance. Dans ces conditions, plusieurs stratégies sont applicables et dépendent du composant concerné. Dans le cadre d'une machine maitrisée (typiquement un *laptop*), ce genre de situation problématique ne devrait néanmoins pas se produire et la mesure de sécurité à adopter est d'empêcher l'exécution du code fautif et d'interrompre la procédure de démarrage.

Le *Secure Boot* présente l'avantage d'une certaine souplesse, notamment dans la gestion des mises à jour, dans sa mise en œuvre. Cette souplesse peut malheureusement se retrouver contreproductive. Il n'est ainsi pas assuré que le *Secure Boot* soit assez finement paramétrable [4]. De plus, si le support du *Secure Boot* est de plus en plus important, il reste un nombre non négligeable d'implémentation UEFI sans *Secure Boot*.

1.4 *Trusted Boot*

Une autre implémentation possible du paradigme d'informatique de confiance est le *Trusted Boot*. Le *Trusted Boot* a été proposé et standardisé par le *Trusted Computing Group* (TCG) [5] en XXXX [6].

Le *Trusted Boot* propose plusieurs fonctionnalités, dont trois qui nous intéresse particulièrement dans le cadre de la sécurisation du démarrage d'une plateforme CLIP :

- La **mesure** du code (*measurement*) ;
- La **sauvegarde** de ces mesures (*reporting*) dans des registres ;
- Le **scellement** conditionné par les valeurs de ces registres de données critiques (*storage*) dont on veut prévenir des accès d'un poste compromis.

Contrairement au *Secure Boot*, qui sanctionne directement la mesure d'un code en décidant s'il est ou non de confiance, le fonctionnement du *Trusted Boot* est plus indirect. Chaque composant logiciel est mesuré au moment de son chargement, mais plutôt que de vérifier cette mesure, elle est sauvegardée dans divers registres. La valeur de ces registres à un instant t conditionnera la libération de données critiques préalablement scellées. Un exemple typique d'utilisation de ce mécanisme de scellement conditionnel concerne la clef de chiffrement d'un disque dur sensible. En conditionnant la libération de la clef de chiffrement d'un disque dur selon la valeur des registres de sauvegarde des mesures, on peut s'assurer que seul un poste non compromis [7] peut déchiffrer le contenu du-dit disque dur. En effet, un poste dans l'incapacité de récupérer la clef de déchiffrement a forcément des registres de sauvegarde de mesure incorrects et cette erreur ne peut s'expliquer que par le chargement d'un code arbitraire.

Grâce à ces registres de sauvegarde, on a une notion beaucoup plus fine de ce qu'est un « code de confiance ». De plus, les mesures ne demandent pas un matériel cryptographique complexe.

Néanmoins, la procédure devient plus complexe. Si l'on a toujours besoin d'une racine de confiance pour la mesure (qui reste les premières fonctions du BIOS), deux nouveaux composants deviennent nécessaires :

- Une racine de confiance pour la sauvegarde des mesures ;
- Une racine de confiance pour le scellement des données critiques.

Le *Trusted Boot* repose en grande partie sur le *Trusted Platform Module* (TPM), un micro-contrôleur standardisé lui-aussi par le TCG. Le TPM intervient dans le *Trusted Boot* en servant de racine de confiance pour la sauvegarde des mesures et de racine de confiance pour le scellement des données critiques.

3. Signature incorrecte, certificat absent de la base de données, etc.
4. Par exemple, certains BIOS pourraient fournir des certificats par défaut impossible à supprimer.
5. Un consortium contenant la plupart des acteurs majeurs du marché de l'informatique, comme AMD, Intel, HPou Microsoft.
6. **Attention** : *Trusted Boot* (plus couramment appelé *tboot*) est aussi le nom d'un chargeur d'amorçage développé par le TCG. Lorsque nous parlons de *Trusted Boot*, nous parlons du paradigme et non pas d'une quelconque implémentation.
7. C'est-à-dire ayant exécuté uniquement du code de confiance

Partition	Description	Chiffrée	Partagée [a]
`/boot`	Chargeur d'amorçage, *initrd*, noyaux CLIP. Partition d'amorçage.	Non	Oui
`/home`	Données des utilisateurs	Oui	Oui
`/var/log`	Journaux systèmes	?	Oui
`/`	Partition racine du système	?	Non
SWAP	Partition de SWAP	Oui [b]	Oui
`/mounts`	?	?	Non
`/vservers/rm_b`	?	?	Non
`/vservers/rm_h`	?	?	Non

a. Les partitions qui ne sont pas partagées sont dédoublées : une par système CLIP installé.
b. Une nouvelle clef est tirée à chaque démarrage du poste CLIP, ce qui empêche la lecture de la SWAP après extinction de la machine

TABLE 1 – Résumé de l'organisation du disque dur d'un poste CLIP

Un TPM possède un certain nombre de registres de configuration nommés *Platform Configuration Register* (PCR). Dans ce document, on notera désormais l'extention du i^{me} PCR par la mesure m : $PCR_i \leftarrow m$. La nouvelle valeur de PCR_i est calculée de la façon suivante :

$$PCR_i := mac(PCR_i||m)$$

Il n'existe que deux façons de modifier un PCR : par extension par une mesure m ou en les réinitialisant par un redémarage de la machine. L'avantage de cette méthode est que les PCR ne reflettent pas uniquement la dernière mesure soumise, mais bien l'historique de ces valeurs. Ainsi, si un code malveillant est exécuté après avoir été mesuré, il ne sera pas en mesure « d'effacer » sa présence du PCR qu'il aura « contaminé ». C'est ainsi qu'il peut remplir son rôle pour la sauvegarde des mesures.

Grâce à un TPM, il est possible de sceller une donnée (typiquement, un fichier) et de conditionner son déchiffrement à des valeurs de PCR. C'est ainsi qu'il peut remplir son rôle de chaine de confiance pour le scellement.

Dans le cadre d'un *Trusted Boot*, les différentes valeurs des PCR au cours du temps doivent être prédictibles. C'est justement en comparant ces valeurs *attendues* avec les valeurs *effectives* qu'il est possible de déterminer si la machine considérée est dans un état de confiance ou dans un état compromis. Cette comparaison est effectuée de manière implicitement lors d'une opération de libération d'une donnée scellée.

2 Intégrité de la partition système dans CLIP

2.1 Chiffrement dans CLIP

On trouvera sur un même poste CLIP deux versions du système d'exploitation. La première est la version la plus récente de CLIP, la seconde est la version précédente. Cette façon de faire permet de continuer à pouvoir utiliser son poste CLIP même en cas de mise à jour défectueuse. L'organisation du disque dur sur lequel est installé CLIP reflète cette dualité. Le Tableau 1 rappelle l'organisation du disque dur d'un poste CLIP et le rôle fonctionnel de chaque partition. Les partitions marquées comme étant partagées sont communes aux deux installations, cela veut dire que peu importe la version CLIP utilisée, ces partitions seront montées, potentiellement modifiées et ces modifications impacteront donc l'autre version de la même façon. La partition `/home` est ainsi partagée pour que l'utilisateur ait accès à ses fichiers peu importe la version CLIP qu'il décide d'utiliser au démarrage.

#	Convention
0	BIOS ROM et flash
1	Configuration du *chipset*
2	PCI ROMs
3	PCI Config
4	Chargeur d'amorçage
5	Configuration du chargeur d'amorçage
6	Ligne de commande du kernel
7	Kernel
8	Initrd

TABLE 2 – Conventions d'usage des PCR d'un TPM

Le chiffrement et le déchiffrement des partitions d'un poste CLIP est réalisée à partir d'une clef primaire sauvegardée dans le fichier `master_key` dans la partition `/boot`. Cette clef peut être protégée à l'aide d'une phrase secrète (*passphrase* en anglais) qui sera demandé à l'utilisateur lors de la séquence de démarrage, mais cette sécurité supplémentaire n'est pas mise en place dans la procédure d'installation courante de postes CLIP pour le personnel de l'ANSSI. Ce choix a été motivé par la volonté de faire un compromis entre sécurité et confort d'utilisation : les besoins en confidentialité ne sont pas les mêmes pour chaque partition et il est acceptable que certaines d'entre elles soient déchiffrées facilement [8]. À l'inverse, les données utilisateurs sont doublement chiffrées : une première fois avec une clef dérivée de la clef primaire et une seconde fois avec une clef dérivée de leur mot de passe. De double chiffrement assure la confidentialité des données de chaque utilisateur de manière robuste.

2.2 Objectifs

L'objectif principal du durcissement de la séquence de démarrage par l'utilisation du TPM est de protéger l'utilisateur contre la compromission de son poste CLIP en liant le bon déroulement des partitions chiffrées d'un poste CLIP à l'intégrité de sa partition `/boot`. Pour ce faire, la solution proposée est de **sceller conditionnellement aux valeurs attendues d'un sous-ensemble des PCR à la fin du boot** la clef primaire de chiffrement (le fichier `master_key` présent dans la partition `/boot`).

Le TCG a édité plusieurs spécifications successives du TPM. La version la plus répandue est celle du TPM v1.2 ; **il s'agit de celle que sera supporté par CLIP**.

Le Tableau 2 donne un résumé des différents rôles attribués à chaque PCR. Les quatre premiers PCR (PCR_0 à PCR_3) sont mis à jour par le BIOS pendant les premiers temps de la séquence de démarrage.

2.3 Procédure d'installation

Les modifications à apporter à la procédure d'installation standard de CLIP pour permettre l'utilisation du TPM ne sont pas lourdes. La création des partitions chiffrées d'un système CLIP est prise en charge par la fonction `map_partition` du script `init_partitions.sh` (présent dans le package `clip-livecd`). Ce script est lui-même logiquement appelé par le script maître de l'installation d'un système CLIP : `full_install.sh` (du même package).

Les étapes supplémentaires nécessaire à réaliser une fois la création des partitions chiffrées sont :

8. On peut dès lors se demander l'utilité de les chiffrer. Cela s'avère utile lorsque l'on veut recycler le poste CLIP : il suffit d'effacer la partition `/boot` pour rendre impossible l'exploitation du disque dur tout entier.

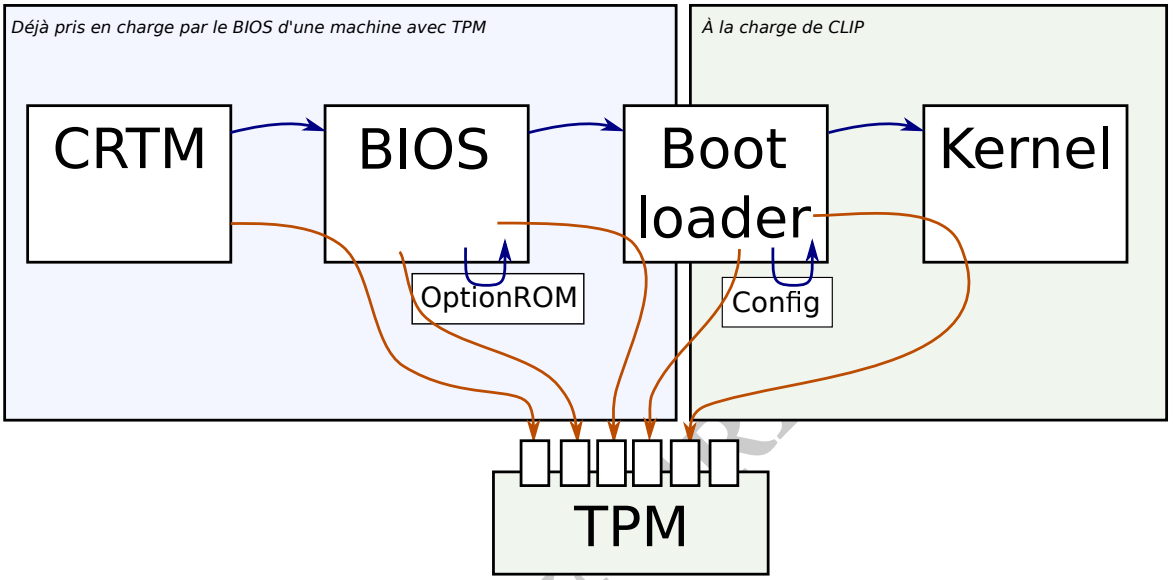


FIGURE 2 – Schématisation de la séquence de démarrage, de la CRTM jusqu'au Kernel

1. Lecture des PCR 0 à 3, spécifiques à chaque machine et imprévisible par l'installeur CLIP (voir Tableau 2) ;
2. Calcul des valeurs attendues par les PCR 4 à 8 (voir Tableau 2) ;
3. Scellement deux fois du fichier `master_key` : création des fichiers `master_key.5.sealed` et `master_key.10.sealed` et suppression du fichier en clair `master_key` ;
4. Ajout d'une autre clef pour le déchiffrement des partitions du système CLIP [9] pour permettre une procédure de récupération (voir sous-section 2.7).

2.4 Procédure de démarrage

La Figure 2 rappelle et illustre le principe de chaîne de confiance de l'informatique de confiance appliquée grâce à l'utilisation d'un TPM. Mis à part la racine de confiance pour les mesures (la CRTM), tous les éléments de la chaîne de confiance sont mesurés par l'élément précédent qui stocke cette mesure dans un registre de configuration du TPM (un PCR) [10]. Une fois mesurés, les composants de la séquence de démarrage mesurés, ils peuvent être exécutés [11].

L'établissement de la chaîne de confiance est primordial et critique pour protéger l'intégrité de la séquence de démarrage. Néanmoins, elle n'est pas uniquement de la responsabilité du système CLIP. Dans un poste disposant d'un TPM, il est de la responsabilité de son micrologiciel [12] d'initier et de constuire la chaîne de confiance, de la racine de confiance jusqu'au chargeur d'amorçage. Par la suite, le chargeur d'amorçage et le noyau du système d'exploitation doivent être modifiés pour « prendre le relais ».

Il existe deux grandes « familles » de micrologiciels :
- Historiquement la plus ancienne, les BIOS (système basique d'entrée/sortie, *Basic Input Output System*) ;
- La plus récentes, les implémentations de la norme UEFI (*Unified Extensible Firmware Inteface*).

9. Le standard LUKS (*Linux Unified Key Setup*) permet de réaliser aisément ce genre d'opérations,
10. Les flèches oranges de la Figure 2.
11. Les flèches bleues de la Figure 2.
12. *Firmware*, en anglais

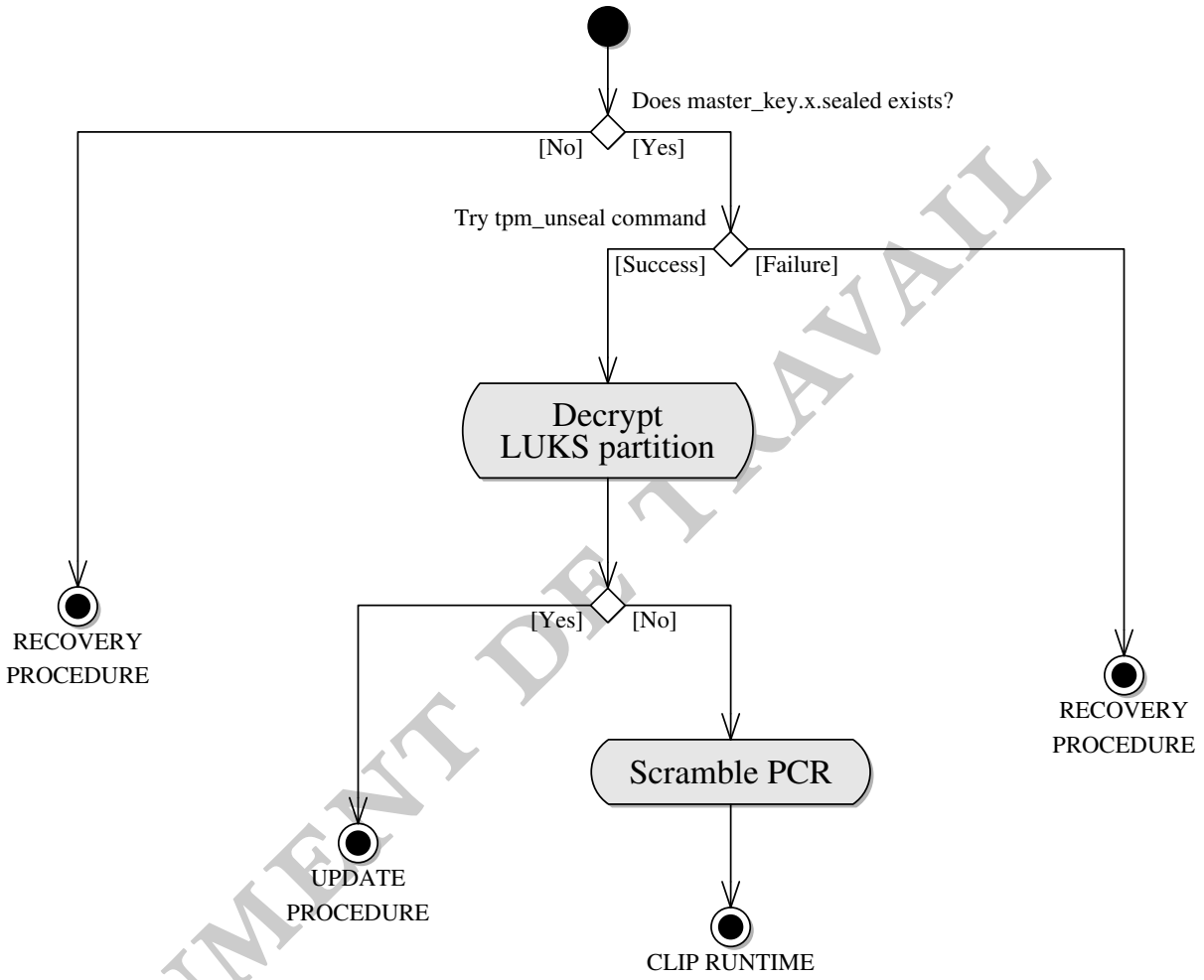


FIGURE 3 – Récupération de la clef primaire et déchiffrement des partitions CLIP

.

Les BIOS et les implémentations UEFI n'utilisent pas du tout la même approche pour gérer démarrage d'une machine et de son système d'exploitation. Actuellement, CLIP n'est compatible qu'avec l'approche du BIOS [13]. Plus d'informations sur la séquence de démarrage dans CLIP peuvent être trouvées dans .

2.5 Procédure de déchiffrement des partitions LUKS

La Figure 3 résume le dérouelemnt de la procédure de déchiffrement des partitions LUKS d'un système CLIP. Cette procédure a lieu pendant l'exécution de l'`initrd` du système CLIP choisi. Elle présuppose donc que la procédure de démarrage s'est correctement déroulé [14].

La première étape est de vérifier l'existence du fichier `master_key.x.sealed` [15]. Si ce fichier n'existe pas, alors le déchiffrement est impossible et une procédure de récupération commence (voir sous-section 2.7). Si le fichier existe, alors il est possible de tenter de le desceller. La réussite ou l'échec de cette étape dépend de

13. Il est néanmoins possible d'installer CLIP sur des plateformes UEFI, mais ces dernières doivent supporter un mode *legacy* leur permettant de se comporter comme un BIOS.

14. Ce n'est néanmoins qu'à la fin de cette procédure de déchiffrement --- et absolument pas avant --- qu'il est possible de déterminer si aucun composant malveillant n'a été chargé.

15. x est à remplacé par 5 ou 10 en fonction du système CLIP choisi par l'utilisateur au démarrage.

la valeur des PCR du TPM. En cas d'échec, une procédure de récupération peut commencer (voir sous-section 2.7). Sinon, la clef primaire est disponible en clair et les partitions sont déchiffrées.

Une fois le disque déchiffré, il est important de vérifier si une mise à jour des paquetages essentiels de CLIP est en attente. Les packages `sys-kernel/clip-kernel` (noyau CLIP) et `sys-boot/syslinux` (chargeur d'amorçage `syslinux` utilisé par CLIP) sont en effet susceptibles d'être modifiés par telle mise à jour et il convient dès lors de lancer une procédure de mise à jour (voir sous-section 2.6) pour s'assurer que la clef primaire puisse toujours été récupérée au prochain démarrage. En effet, une modification des packages `sys-kernel/clip-kernel` et `sys-boot/syslinux` entraînent une modification des fichiers mesurés pendant la procédure de démarrage, ce qui a un impact sur la valeur finale des PCR.

Dans le cas où aucune mise à jour n'est en attente, alors la procédure de déchiffrement prend fin et la séquence de démarrage peut correctement se terminer. Pour protéger la confidentialité de la clef primaire, il convient néanmoins de brouiller les PCR le plus tôt possible après avoir déterminé que leurs bonnes valeurs n'étaient plus nécessaires.

2.6 Procédure de mise à jour

2.7 Procédure de récupération

3 Difficultés de mise en œuvre

3.1 UEFI ou BIOS Legacy ?

3.2 Choix et modification du chargeur d'amorçage

Références

[evil09] *Why do I miss Microsoft BitLocker ?*, Joanna Rutkowska, January 2009.